VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Tvorba testovacího malware
Design of Testing Malware

2014 **Filip Zatloukal**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Filip Zatloukal**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Tvorba testovacího malware
Design of Testing Malware

Zásady pro vypracování:

Práce se zabývá analýzou, detekcí a tvorbou testovacího malware. Cílem je vytvořit testovací programy, které by se chovaly jako škodlivý malware a to pro studijní účely.

1. Seznamte se s principy škodlivého kódu.
2. Navrhněte vlastní implementaci testovacího malvare se zameřením na počítačové viry.
3. Určete vhodné testovací postupy a studie.
4. Vytvořte základní dokumentaci.
5. Diskutujte dosažené výsledky a navrhněte možná vylepšení.

Seznam doporučené odborné literatury:

[1] Merhaut F., Zelinka I., Úvod do počítačové bezpečnosti, Fakulta aplikované informatiky, UTB ve Zlíně, Zlín, 2009
[2] Peter Szor, Počítačové viry - analýza útoku a obrana, Zoner Press
[3] Pokorný J., Hacking - umění exploitace, Zoner Press
[4] Lance J., Phishing bez záhad, Grada 2007

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **prof. Ing. Ivan Zelinka, Ph.D.**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Poděkování

Tato práce byla vypracována s podporou projektu Rozvoj lidských zdrojů ve výzkumu a vývoji moderních soft computingových metod a jejich praktického využití, reg. č. CZ.1.07/2.3.00/20.0072 podpořeného Operačním programem Vzdělávání pro konkurenceschopnost, financovaného ze strukturálních fondů EU a státního rozpočtu ČR.

Acknowledgement

This thesis was prepared with the support of the Development of human resources in research and development of latest soft computing methods and their application in practice project, reg. no. CZ.1.07/2.3.00/20.0072 funded by Operational Programme Education for Competitiveness, co-financed by ESF and state budget of the Czech Republic.

Prohlášení

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě, dne 10. dubna 2014

Filip Zatloukal

Abstrakt

Diplomová práce je zaměřena na téma počítačové bezpečnosti, především v oblasti škodlivého kódu. Podává ucelený přehled o různých typech malware, principech infekce, šíření, ukrytí a následného vykonání kódu. Cílem je také vytvořit funkční program, který je možno klasifikovat jako malware. Implementace slouží pouze pro studijní účely a v praxi ověřuje teoretické poznatky. Program není šířen, jsou ovšem diskutovány možnosti jeho rozšíření a možné škody jim způsobené. Autor se následně zaměřuje na testování svého produktu a hodnocení dosažených výsledků. Dle výsledků je navrhnuta forma detekce, prevence a obrany proti hrozbě.

Klíčová slova

malware, škodlivý kód, počítačová bezpečnost, detekce malware

Abstract

The thesis is focused on computer security, particularly in the area of malicious code. It gives a comprehensive overview of the different types of malware, principles of infection, dissemination, hiding and subsequent execution of code. The aim is to create a functional program which can be classified as malware. The implementation is used only for educational purposes and practical tests of theoretical knowledge. The program is not distributed, however, possibilities for expansion and possible damage caused by it is discussed. The author then focuses on testing the product and assessment of results. Based on the results, detection, prevention and defense against the threat is suggested.

Keywords

malware, malicious code, computer security, malware detection

Seznam použitých symbolů a zkratek

ADS	alternate data streams
AP	access point (přístupový bod)
API	application programming interface
ARP	address resolution protocol
BSS	block started by symbol
DNS	domain name systém
DoS	denial of service
HFS	hierarchical file system
HTTP	hypertext transfer protocol
HTTPS	hypertext transfer protocol secure
IP	internet protocol
ISO	international organization for standardization
JRE	Java runtime environment
NAT	network address translation
NTFS	new technology file system
OS	operační systém
OSI	open systems interconnection model
PE	portable executable
TCP	transmission control protocol
TZ	trestní zákoník
VPN	virtual private network

Obsah

1Úvod....10
1.1Trestní zákon České republiky ve vztahu k malware....11
1.2Počítačová kriminalita a historie....11
1.2.1I love you (2000)....12
1.2.2Code Red/Code Red II (2001)....12
1.2.3Slammer/Sapphire (2003)....12
1.2.4Sobig (2003)....12
1.2.5Mydoom (2004)....13
2Teoretická část....14
2.1Motiv útočníka....14
2.2 Definice a struktura malware....14
2.3Rozdělení malware....15
2.3.1Počítačový červ....15
2.3.2Počítačový virus....15
2.3.3Trojští koně....16
2.3.4Spyware....16
2.3.5Adware....16
2.4Důležitá příprava....17
2.5Sběr informací....17
2.5.1Informace o vzdáleném systému skrze vnitřní síť....18
2.5.2Informace o vzdáleném systému skrze vnější síť....19
2.5.3Sociální inženýrství....20
2.6Cílový systém....23
2.6.1Strojový kód....24
2.6.2Operační systém....24
2.6.3Běhová prostředí nad operačním systémem....25
2.7Tvorba malware....26
2.7.1Programování....26
2.7.1.1Procesor a paměť....27
2.7.2Akce....28
2.7.3Lokální exploit....29
2.7.3.1Přetečení zásobníku....29
2.7.3.2Přetečení na haldě....29
2.7.3.3Přetečení v segmentu bss....30
2.7.4Ukrytí....30
2.7.4.1Ukrytí před uživatelem....30
2.7.4.2Ukrytí před bezpečnostním software....31
2.7.5Šíření....32
2.7.5.1Šíření na lokálním systému....32
2.7.5.2Šíření na vzdálené systémy....32
2.7.6Komunikace....33
2.8 Penetrace....34

2.8.1 Fyzicky přístupný stroj....34
2.8.2Vzdálený exploit....34
2.8.3Získání přihlašovacích údajů....35
2.8.4Metody sociálního inženýrství....35
2.9Životní cyklus....36
3Praktická část....37
3.1Virové generátory....37
3.1.1Virus Construction Set (1990)....37
3.1.2VBS Worm Generator (2000)....37
3.1.3Senna Spy Internet Worm Generator (2000)....37
3.1.4NGVCK (2001)....37
3.1.5Novější generátory....38
3.2Cíle praktické části....38
3.2.1Požadavky na malware....38
3.2.2Kritéria a hodnocení....38
3.3Příprava....39
3.4Struktura malware....39
3.5Cílové operační systémy....40
3.6Programování....40
3.6.1Implementace malware....40
3.6.1.1Po spuštění....41
3.6.1.2Zvýšení oprávnění....42
3.6.1.3Komunikace....42
3.6.1.4Protokol....43
3.6.1.5Akce....44
3.6.1.6Penetrace a šíření....44
3.6.2Implementace serverové části....45
3.6.2.1Principy fungování....45
3.6.2.2Komunikace....46
3.6.2.3Protokol....46
4Dosažené výsledky a studie....48
4.1Testování....48
4.2Životní cyklus a možné následky rozšíření....48
4.3Budoucí vylepšení a vývoj....48
4.4Detekce malware....49
4.5Prevence a doporučení....50
4.6Hodnocení....50
5Závěr....52

"Jediný opravdu zabezpečený systém je ten, který je vypnutý, zasazený v bloku betonu, zavřený v místnosti potažené olovem a střežené ozbrojenou stráží."

Gene Spafford, profesor počítačových věd - Purdue University.

Prohlášení autora

Autor práce prohlašuje, že veškeré části tohoto projektu (textové i digitální) slouží výhradně ke studijním účelům. Práce a poznatky nebyly a nebudou autorem užity k ilegálním činnostem, či je jakoukoli formou podporovat.

Autor se zříká odpovědnosti, a to v plném rozsahu, v každé situaci, kdy je práce, či kterákoli její část, užita jinou osobou.

1 Úvod

Žijeme v moderním světě, kde nás technologie obklopují na každém kroku. Chytrý telefon obsahující operační systém Vás ráno probudí, ukáže aktuální zprávy a předpověď počasí. Cestou do práce ani nevnímáte technologie, kterými je přeplněno Vaše auto. Za to v zaměstnání si jimi rádi ulehčíte těžkou práci nebo naopak jste nuceni je používat a trávit čas před obrazovkou počítače. Člověk 21. století si bez technologií nedokáže představit život.

Moderní doba směřuje stále více ke globalizaci a vzájemné konektivitě. Objevují se další možnosti připojení - je běžnou praxí mít svůj telefon stále připojený k síti mobilního operátora a využívat datové konektivity. Kromě budov se k bezdrátové sítí můžete připojit i ve vlacích, hromadné dopravě a brzy i v letadlech. Doba jde stále dál a každým dnem se na celosvětovou síť internet napojí nová a nová zařízení - počínaje počítači a tablety přes telefony, čtečky knih, televizory a konče hodinkami, brýlemi či jiným nositelným zařízením. V nejbližších letech budou připojeny Vaše domy, auta, pračky, mixéry, příbory ... a nakonec třeba i Vy sami.

Vše pozitivní má ale i svou odvrácenou tvář. Rozebírat všechna možná negativa by vydalo na samostatnou, filozoficky zaměřenou, práci. Čtenář proto bude zasvěcen do jednoho významného a velmi rozsáhlého tématu, které vývoj technologií otvírá. Počítačová bezpečnost a především pak škodlivý kód (označován také jako malware). Každý člověk je nedokonalý a právě to se odráží i na technologiích jako jsou počítačové systémy. Čím více se svět propojuje, a systémy se stávají rozsáhlejšími, tím více možností získávají i pachatelé. Poznamenejme, že škodlivé kódy nejsou směrovány jen na počítače jako takové, ale i na další elektronická zařízení, která mohou být napadnutelná. Čtenář by si měl uvědomit, že i tato zařízení mohou zasahovat do jeho soukromí a nést osobní či cenná data. Nevyjímaje bankovní počítačové systémy, které schraňují také Vaše finance.

Je patrné, že neoprávněná manipulace s takovými systémy v globálním měřítku může poškodit jednotlivce, skupiny, instituce i celé státy. Výrazně více pocítíte důsledky jste-li cílem pachatele právě Vy a je-li útok úspěšný. Tyto podněty mne přiměly k dalšímu přemýšlení: "Jak snadno se útok může stát úspěšným? Jakým způsobem mne může útočník v digitálním světě napadnout a jaké mohou být opravdové důsledky? Jsou systémy dobře chráněny a jak co nejvíce eliminovat rizika?"

Otázky ve mne evokovaly zájem o tuto problematiku. V projektu se proto vžiji do role útočníka, abych mohl lépe pochopit taktiky pachatele a získané poznatky využít k lepší prevenci a detekci útoků. Vědomosti dále využiji ke zkvalitnění zabezpečení mnou spravovaných systémů a vytvořených programů. Teoretické i praktické informace budou v této prací sdíleny i těm, kteří se chtějí podílet na zvýšení bezpečnosti počítačových systémů.

1.1 Trestní zákon České republiky ve vztahu k malware

Trestní činnost v oblasti výpočetní techniky se nazývá "počítačová kriminalita". 1. ledna 2010 nabyl účinnosti na území České republiky nový trestní zákoník, zákona č. 40/2009 Sb. nahrazující předešlý trestní zákoník č. 140/1961 Sb. Všechna vyjádření této podkapitoly se budou vztahovat k tomuto novému TZ, nebude-li řečeno jinak, a budou směrována k tématu počítačové kriminality.

Do oblasti zabývající se počítačovou bezpečností, osobními údaji či manipulací s informacemi mohou být zařazeny tyto paragrafy:

- § 181 Poškození cizích práv - vztah k phishingu, hoaxu či nevyžádané pošty (spam).
- § 182 Porušení tajemství dopravovaných zpráv - pojednává mimo jiné i o elektronické komunikaci.
- § 230 Neoprávněný přístup k počítačovému systému a nosiči informací - paragraf zabývající se neoprávněným vniknutím do cizího počítačového systému se může aplikovat také na penetraci pomocí malware. Zákon vymezuje odnětí svobody až do výše osmi let, zákazem činnosti nebo propadnutím věci či majetkové hodnoty a to dle závažnosti spáchaného činu.
- § 231 Opatření a přechovávání přístupového zařízení a hesla k počítačovému systému a jiných takových dat - zahrnuje jakékoliv opatření si, uchovávání či distribuci přístupových metod do cizího systému. Vychází z § 182 a § 230
- § 232 Poškození záznamu v počítačovém systému a na nosiči informací a zásah do vybavení počítače z nedbalosti - poukazuje na porušení počítačových systémů z nedbalosti a stanovuje dobu odnětí svobody až na dva roky.

TZ specifikuje i jiné trestní činnosti počítačové kriminality, ty zde ovšem nejsou jmenovány, jelikož přímo nekorespondují s tématem práce.

§ 14 TZ rozděluje trestné činy na přečiny a zločiny - dle jejich závažnosti. Je zřejmé, že úmyslným užití malware a následným vniknutím do cizího systému je zařazeno do kategorie zločinů. Způsobí-li pachatel svým úmyslným jednáním vysokou škodu, může být odsouzen odnětím svobody až do výše osmi let, peněžním, či jiným trestem.

Je důležité si uvědomit, že již samotným přechováváním programu umožňujícího vniknutí do cizího systému je porušován zákon (§ 231) a držitel může být potrestán odnětím svobody v maximální výši jednoho roku, peněžním, či jiným trestem.

1.2 Počítačová kriminalita a historie

"Počítačovou kriminalitu je třeba chápat jako specifickou trestnou činnost, kterou je možné

spáchat pouze s pomocí výpočetní techniky, a kde je výpočetní technika předmětem trestného činu nebo pachatelovým nástrojem ke spáchání trestného činu."[15] Malware je tedy nedílnou součástí počítačové kriminality. V roce 2008, dle firmy Symantec Corporation, přesáhl počet různých exemplářů malware hranici jednoho miliónu[16]. Ze zprávy dále vyplývá, že celé dvě třetiny nových hrozeb byly vytvořeny během roku 2007.

V dalším výzkumu firma prezentuje nejvíce rozšířený malware v historii[17]:

1.2.1 I love you (2000)

Počítačový červ zaměřený na systémy Windows, vytvořený v jazyce VBScript a šířený pomocí elektronické pošty. K jeho aktivaci je potřeba spuštění ze strany uživatele. Po spuštění vyhledává emailové adresy aplikace Microsoft Outlook a na nalezené kontakty se rozešle jako příloha, dále přepisuje uživatelské soubory a vytváří záznamy v systémových registrech.

Později bylo vytvořeno i množství modifikací tohoto červa.

1.2.2 Code Red/Code Red II (2001)

Červ zneužívající vzdálené zranitelnosti MS01-033 v systému Microsoft IIS web server. Po aktivaci proběhlo skenování dalších IP adres a také šíření na takto postižené stroje. Červ byl použit např. k útoku typu denial of service, při kterém byly, mimo jiné, postiženy i služby domény whitehouse.gov.

1.2.3 Slammer/Sapphire (2003)

Červ naprogramovaný v assembleru, napadající zranitelnost MS02-039 produktu Microsoft SQL Server 2000 či jiného programu obsahující Microsoft SQL 2000 Desktop Engine. Šíří se výhradně přes síť, na vzdáleném systému využije chyby přetečení zásobníku a usídlí se pouze v operační paměti (na disk se neukládá). Po spuštění náhodně generuje další IP adresy, na které se snaží rozšířit - tím generuje značný internetový provoz.

1.2.4 Sobig (2003)

Jedná se o počítačového červa šířeného pomocí přílohy elektronické pošty. Existuje v šesti variantách, nejvíce se rozšířila varianta Sobig.F. Obsahuje vlastní SMTP engine a není tedy závislý na konkrétním poštovním klientovi. Červ musí být spuštěn uživatelem, poté se nahraje na disk a začíná prohledávat místní soubory a vyhledávat v nich řetězce podobné emailovým adresám. Mezi další vlastnosti patří šíření pomocí síťového sdílení, vytvoření zadních vrátek systému, či stahování instrukcí k dalšímu šíření. Tento malware byl navrhnut k ilegálním komerčním účelům - vytvoření sítě botnet a následného šíření spamu.

1.2.5 Mydoom (2004)

Červ cílený na platformu Microsoft Windows vytváří po spuštění zadní vrátka do systému a snaží se šířit sám sebe na další stroje. Jeho varianty Mydoom.A a Mydoom.B jsou šířeny přes elektronickou poštu a P2P sítě- v příloze pošty je zasílán v různých formátech (s různými příponami). Je známa také varianta Mydoom.C, která se snaží využít již existujících zadních vrátek systému, vytvořených předchozí variantou tohoto červa a provést aktualizaci. Mydoom.B brání v přístupu na webové stránky firmy Microsoft a známých antivirových společností.

Název	Počet infekcí	Způsobená škoda
I love you	500 000	15 000 000 000$
Code Red	1 000 000	2 600 000 000$
Slammer	200 000	1 200 000 000$
Sobig.f	2 000 000	37 100 000 000$
Mydoom	2 000 000	38 000 000 000$

Tento výčet zdaleka není kompletní, ale je dobrou ukázkou rozmanitostí různých malware a škod jimi napáchaných.

2 Teoretická část

2.1 Motiv útočníka

Ptát se "Proč lidé vytváří počítačové viry?" je stejné jako otázka "Proč lidé páchají zločiny?"[18]. V širším slova smyslu nemusíme vždy najít odpověď, zaměříme-li se však na jednotlivce či skupiny, můžeme ze známých případů odvodit záměry, se kterými útočníci malware tvoří.

Většinu úmyslů zahrnuje tento seznam[19]:

- Krádež citlivých informací - útočník se snaží získat údaje o platebních kartách, heslech, osobách, společnostech či ukrást citlivá data v podobě datových souborů. Ukradené informace může použít buď pro své vlastní potřeby či prodat jiné osobě, popřípadě nabídnout na černém trhu.
- Generování zisku - tvůrce může využít napadené počítače k rozesílání nevyžádané pošty či pronajímat síť napadených počítačů. Ty pak mohou být nájemcem užity rovněž k rozesílání pošty, dále k provádění útoků, především typu denial of service a crackování hesel hrubou silou.
- Převzetí kontroly nad zařízením - útočník může chtít pouze získat přístup k určitému zařízení. Buď z důvodů osobních (pomsta, vydírání, atd.), či na zakázku.
- Poškození - cílem může být také pouhé poškození systému, sítě, či poškození společnosti, na niž je útok směřován.
- Důkaz schopností - předvést své schopnosti ostatním může být dalším z motivů útočníka. Malware stvořený pro tento účel nemusí nést nebezpečný náklad, ale může obětem zobrazovat nevyžádaná sdělení (hlášení systému, vtipné animace či jiný kreativní obsah).

2.2 Definice a struktura malware

"Malware je souhrnné označení pro potenciálně nebezpečný či přímo škodlivý software. Pochází z anglického MALicious softWARE. Zahrnuje nepřeberné množství rizikových aplikací, jako jsou viry, červi (worms), trojští koně, spyware, rootkity apod."[20]

Malware nemá vždy jednoznačnou strukturu, protože ta je závislá na typu systému, pod kterým bude malware provozován. Obecně se však jedná o proveditelný kód nebo taková data, která způsobí po svém načtení (nadřazeným systémem) útočníkem zamýšlenou akci. Existují také typy malware, které využívají chyb autorů cílového systému ke své vlastní činnosti.

Lze tedy uvést jen obecnou strukturu. Ne všechen malware obsahuje tyto části, jiný typ malware

může naopak obsahovat části zde neuvedené:

- Hlavička - některé typy souborů potřebují ke svému spuštění hlavičku, obsahující metainformace potřebné k dalším úkonům. Příkladem může být PE hlavička proveditelných souborů Microsoft Windows.
- Tělo - vlastní tělo počítačového červa, viru či trojského koně. Obsahuje samotný kód, který má malware provádět. Dle typu může být v binární či textové podobě.
- Náklad (anglicky "payload") - útočníkem definovaná data, která budou nahrána do systému po úspěšném spuštění malware.
- Dekryptor - v rámci maskování a ochrany před antiviry může být tělo viru zašifrované. Dekryptor se aktivuje po spuštění viru a většinou v reálném čase provádí dešifrování jeho těla. V případě polymorfních virů také může smysluplně zaměňovat pořadí prováděných instrukcí.

2.3 Rozdělení malware

Malware se dělí do různých kategorií - dle účelu, způsobu šíření a penetrace, dle cílového hostitele a dalších kritérií.

Dělení dle účelu:

- Nakažlivý malware - počítačoví červi a viry
- Utajení - trojští koně
- Sledování - spyware
- Reklamní aplikace - adware

Dále jmenované typy nákaz jsou podmnožinou malware[12]:

2.3.1 Počítačový červ

Autonomní software, jenž vytváří kopie sama sebe a ty dále šíří především počítačovou sítí formou síťových paketů (někdy se také pojmem počítačového červa označují i takový malware, který se šíří elektronickou poštou). K infekci dochází ve většině případů přes zranitelnost cílového počítačového systému. Po úspěšném napadení vykoná nejen naprogramované příkazy, ale především se zkusí dále šířit.

2.3.2 Počítačový virus

Název je odvozen od virů biologických, jelikož je svým chováním připomínají. Aby byl virus proveditelný, musí být přeložen do formátu, ve kterém jej bude cílový operační systém schopen

spustit. Virus může být spuštěn i jinou entitou, než operačním systémem.

Podle místa působení dělíme viry dále do kategorií:

- Souborový virus - bývá distribuován ve formě proveditelného kódu. Většinou je autorem předem definována množina operačních systémů, pod kterou bude virus schopen vykonávat svou činnost. Souborový virus pak může své kopie vytvářet, přepisovat jiné soubory (přepisující viry), či se k jiným spustitelným souborům připojovat (parazitické viry).
- Boot viry - napadají master boot záznamy a zaváděcí sektory vyměnitelných médií. Virus bude spuštěn ihned, jakmile mu BIOS předá zavádění systému. Nahraje se do paměti (a stane se paměťově rezidentním), obsadí adresy systémových služeb a samozřejmě zavede původní systém.
- Skriptové viry - viry vytvořené pomocí skriptovacího jazyka daného systému. Je možné využít pouze ty možnosti, které jsou přes skriptovací jazyk přístupné. Bývají uložené jako skript uvnitř dávkového souboru a může být velmi jednoduché tyto viry napsat.
- Makroviry - určité programy umožňují uživateli automatizaci a zjednodušení práce pomocí tzv. "maker". Nejčastěji tuto funkci nabízí kancelářské aplikace. Jsou tedy podobné skriptovým virům, ale zde jej obsluhuje jiný software než operační systém. Makrovir využívá jen funkce nadřazeného programu a cílového makrojazyka.

Viry mohou být uloženy i jiným způsobem - např. jako datové struktury, musí jej ovšem aplikace, pro kterou jsou určeny, spustit, jinak se stávají bezcennými.

2.3.3 Trojští koně

Jsou připojeni jako nežádoucí kód k jinému software, který je pro uživatele žádoucí. Nejsou schopni sebe-replikace a musí být tedy staženi a spuštěni uživatelem. Bez vědomí uživatele ovšem vykonává svou předem definovanou činnost. Nejčastěji vytváří zadní vrátka do systému.

2.3.4 Spyware

Malware zaměřený především ke sledování a získávání dat z cílového systému. Může být naprogramován k reprodukci, ale nemusí. Jeho prioritou je odesílání získaných informací a maximální možnosti utajení.

2.3.5 Adware

Program podporující reklamní či propagační činnost, který je vložen do jiného software produktu. Většinou se jedná o legální kód, který neprovádí žádnou škodlivou činnost. Adware se vyskytuje většinou ve volně dostupných aplikacích a cestou, kdy uživateli zobrazuje reklamu, financuje svůj další rozvoj.

Stává se, že někdy je obtížné zařadit škodlivý program do přesné kategorie. Proto je přirozené, že specifický malware může obsahovat prvky různých kategorií. Některé publikace také uvádí další rozdělení, pro tuto práci jsou ovšem informace dostačující. Kategorizace činností, které malware může provádět, bude popsána později, stejně tak možnosti penetrace, či užití metod sociálního inženýrství.

2.4 Důležitá příprava

Před tvorbou malware si autor zpravidla musí ujasnit záměry a cíle své práce.

- morální a trestní odpovědnost
- časový harmonogram a životnost projektu
- požadované výsledky
- cílová skupina systémů a uživatelů
- možnosti penetrace
- šíření malware
- náklad / vykonávané akce
- utajení
- jiné specifické úkony

Dále pak tvůrce zhodnotí své schopnosti a znalosti, případně potřebné informace nastuduje. Logicky vyplývá, že k naprogramování účinného malware potřebuje autor především poznatky o cílovém systému a dovednosti v algoritmizaci. Znalostem a dovednostem pak přirozeně musí přizpůsobit i svůj výsledný produkt.

2.5 Sběr informací

Vytváření malware se v různých bodech shoduje s hackerskými postupy. Útočník také potřebuje dobře znát svůj cíl a získat o něm, pokud možno, co nejvíce informací. "Prvním krokem hackerského procesu je získávání informací o cíli. Získané informace se nazývají otisky prstů. V době internetu jsou kousky informací dostupné z různých zdrojů"[5].

Záleží ovšem, na jaký cíl se útočník chce zaměřit:

- Specifický cíl - subjektem může být jednotlivec, skupina, společnost či organizace. Tvůrce malware se tedy zaměří na svůj cíl, o kterém se bude snažit získat co nejvíce informací. Především pak o síťové infrastruktuře subjektu, konkrétních systémech, jejich odlišnostech a konfiguracích.

- Obecný cíl - spíše než na konkrétní subjekt se tvůrce zaměří na určitou technologii a systémy. V zásadě útočník neřeší otázku "Kdo systémy užívá?", ale řeší množství uživatelů technologie. To je důležité v situaci, kdy cílem útočníka je napadnout, pokud možno, co nejvíce strojů a vytvořit z nich ilegální síť. Potřebné informace k útoku na obecný cíl: možnosti vykonávání kódu uvnitř napadeného systému, bezpečnostní slabiny, druhy komunikace a jiné.

Další části kapitoly se zaměřují především na získávání informací o konkrétním cíli. Tvůrce malware si musí ověřit, jakým způsobem je jemu cílový systém dostupný:

- Fyzická dosažitelnost - útočník má přímý přístup k zařízení. V tomto případě bývá napadení zpravidla nejjednodušší. Útočník může sám na stroji spustit svůj malware a nechat vykonávat jeho činnost.
- Vzdálená dosažitelnost - výhodou, a současnou slabinou, informačních technologií je možnost vzdáleného přístupu k nim skrze síť. Přístup skrze síť nemusí automaticky znamenat, že je možné se k cíli dostat přímo či nepřímo přes internet - cíl může být přístupný pouze z autonomního systému, ke kterému je připojen. Zda-li je možné se k němu dostat i z internetu záleží na vnitřní infrastruktuře autonomního systému a také na nastaveném směrování, překladu adres a vnitřním zabezpečení.
- Žádná dosažitelnost - existují stanice, které jsou z různých důvodů od sítě úplně odpojeny. To ovšem neznamená, že není možné je jistým způsobem nakazit. Útočník může svůj malware poskytnou na datovém nosiči, který bude jinou osobou otevřen a následně spuštěn. Tato záležitost spadá spíše do oblasti sociálního inženýrství, která bude probírána později. Problém ovšem nastává v případě pokusu o získání dat z takto nepřístupného stroje. Útočník je v této situaci opět závislý na zásahu třetí strany, která má možnost přístupu.

Následující podkapitoly se budou věnovat získávání informací vzdáleně, prostřednictvím sítě. Je potřeba uvést rozdělení, kdy je cíl připojen k vnitřní síti stejně jako útočník, či je útočník připojen přes síť vnější. Druhá z možností zahrnuje také přístup skrze síť internet. V textu je předpokládáno, že veškeré síťové prvky komunikují pomocí modelu TCP/IP.

2.5.1 Informace o vzdáleném systému skrze vnitřní síť

Uvažujme, že cíl je přímo dosažitelný skrze síť (není tedy odstíněn síťovou infrastrukturou). Útočník zajisté bude chtít zjistit co možná nejvíce informací, před penetrací to ovšem bude možné jen v určité míře. Množství vzdáleně zjistitelných informací závisí na konkrétních službách běžících na cílovém stroji. Ne všechny informace budou také k penetraci použitelné.

Nejdříve je zapotřebí znát cílovou IP adresu. IP adresa je používána při komunikaci na 3. vrstvě referenčního modelu IOS/OSI. Bude-li útok probíhat z vnitřní strany sítě, je samozřejmě

podstatné nalézt adresu vnitřní a ne vnější. Nezná-li ji útočník, může využít různých metod, jakými si ji opatřit. Zaslat DNS dotaz, odeslat ARP dotaz, odchytávat komunikaci, vyčíst ji z logovacích souborů uložených při dřívější komunikaci a to buď z lokálního systému, nebo z jiného stroje, se kterým cíl komunikoval. Možností je více, především také využitím sociálního inženýrství (viz dále) - příkladem může být i fiktivní webová aplikace, na kterou je oběť dovedena phishingovými triky.

Následuje skenování portů. "Skenování portů je proces, kdy se snažíme zjistit, jaké síťové služby běží na dané IP adrese"[21]. Většina síťových služeb má předdefinován port, na kterém se v základní konfiguraci spouští. Mnoho administrátorů nechává porty nezměněné, tím ulehčí útočníkovi identifikaci služby. Existují databáze portů[22], kde může každý dohledat aplikaci na tomto portu běžící (v základní konfiguraci). Útočník si tak udělá představu o cílovém systému. Především také o operačním systému, jelikož ne všechny služby jsou podporovány každým systémem. Pachatel se dále může zkoušet připojit na různé služby a hledat další informace skrze ně. Častým případem může být připojení na službu webového serveru. Ze strany provozovatele bývá většinou zapnutý záznam (log), který ukládá všechny zaslané žádosti na tuto službu - datum a čas, zdrojové IP adresy a požadavek samotný.

Útočník má v místní síti výhodu, jelikož se může pokusit provést útok na síťovou infrastrukturu, DNS či jinou službu. Nejvíce zajímavé bude pro pachatele odposlouchávání cizí komunikace - software zachytává rámce/pakety určené pro oběť a ukládá jejich obsah na útočníkův stroj. Odposlech je možný jak v rámci drátové, tak bezdrátové sítě. Bezdrátová síť má v tomto ohledu, z pohledu zabezpečení, nevýhodu, jelikož signál je šířen prostorem a je tedy lehčí jej odchytit, na druhou stranu dnešní přístupové body zvládají signál bezpečně zašifrovat. Odposlech je v aktuálním případě pasivním typem útoku, kdy provoz není měněn.[11]

Aby mohl útočník odposlouchávat na přepínaných sítích, musí zajistit, aby se požadovaný provoz dostal na jeho linku. Nejjednodušší, ale často nejméně proveditelné, je připojit zařízení přímo na páteřní síť, či pomocí habu do požadované části síťové topologie. Některé síťové prvky umožňují v rámci ladícího módu nastavit Switched Port Analyzer a nechat na nastavený port zasílat veškerý provoz. Útočník ale musí získat přístup ke konfiguraci daného prvku.

Jiné možnosti spočívají v provedení aktivního odposlechu - musí být proveden útok na síť, který způsobí odklonění provozu. Nejpoužívanější techniky se nazývají[3][5] ARP cache poisoning, DNS poisoning, MAC flooding, či jiné.

Práce se jimi nebude podrobněji zabývat, jelikož se tematicky jedná o postupy používané spíše v hackerské praxi.

2.5.2 Informace o vzdáleném systému skrze vnější síť

Zpravidla je vždy přístup z vnější sítě více zabezpečený, než přístup ze sítě vnitřní. Cílový systém dokonce nemusí být z vnější sítě přístupný - nachází se za směrovačem, který provádí překlad adres (Network Address Translation, či Network Masquerading) a je chráněn

firewallem. Není-li přímo na hraničním směrovači nastaveno směrování do vnitřní sítě, nebude možné se na vnitřní server z vnější sítě připojit.

Chce-li se útočník dostat do vnitřní sítě, musí nejdříve získat kontrolu nad hraničním směrovačem. Možností je několik. V minulosti se již objevily zranitelnosti, pomocí kterých je možné ovládnout směrovač [23] (včetně chyby krvácejícího srdce) - u některých je ovšem zapotřebí provést útok z vnitřní sítě. Útočník může vyzkoušet připojení do vnitřní infrastruktury pomocí slabě zabezpečené bezdrátové sítě. Další možností je útok hrubou silou na vzdálenou správu zařízení, VPN koncentrátor či pokus o napadení jiné služby. Je-li ve firewallu povoleno připojení na jiný systém uvnitř sítě, může pachatel zaútočit na něj a z napadeného systému provést další útok, tentokrát na vnitřní cíl.

Další útoky pak již počítají s akcí ze strany oběti, která vytvoří spojení. Poměrně novým typem útoku je Drive-By Pharming[13], který po navštívení webové stránky uživatelem spustí JavaScript, jež se snaží připojit k administraci výchozí brány (za pomoci výchozího hesla) a změnit adresy DNS serverů. Chce-li pachatel zaútočit na překlad adres, může využít techniku NAT Pinning[24], kdy oběť navštíví kompromitovanou webovou stránku, ta se snaží otevřít spojení, které se pro směrovač jeví jako spojení jiného než HTTP(S) protokolu a pro které je potřeba otevřít nový lokální port. Směrovač se proto v dobré vůli pokusí využít technologii NAT traversal, pro kterou útočník specifikuje požadovaný port, ten je přesměrován a pachatel jej může využít. Technikou Drive-By Pharming a NAT Pinning lze napadnout pouze směrovače na tyto útoky náchylné.

Nejčastěji dochází k napadení oběti za firewallem opět přes kompromitovanou webovou stránku, která využije zranitelnosti internetového prohlížeče či aktivního modulu. Ke zjištění informací o vzdáleném stroji, který je schován za firewallem, ovšem není potřeba ani nelegální činnosti. Programátor vytvoří webovou stránku, která se po otevření pokusí z cílového spojení, zaslaného HTTP požadavku a webového prohlížeče získat potřebné informace. Ze spojení přečte především IP adresu, z HTTP hlavičky pak mnohem více údajů - užitý operační systém a prohlížeč, včetně jeho verze (User-Agent), předchozí zobrazenou stránku (Referer), jazykovou mutaci (Accept-Language) či jiné. Webový prohlížeč může poskytnout cookies třetích stran, informace skrze JavaScript (počet aktivních modulů, rozměr okna, počet instalovaných písem atd.)[25] a zásuvné moduly. Z těchto otisků si může autor udělat představu o cílovém systému a stává se pro útočníka mnohem lépe identifikovatelným[26].

Zbývá vyřešit, jakým způsobem přiměje pachatel oběť zavítat na takovýto web. Přesměrování provozu vzdáleného cíle je v prostředí internetu velmi problematické, proto bývá častým scénářem použití metod sociálního inženýrství.

2.5.3 Sociální inženýrství

"Sociální inženýrství užívá manipulaci, vliv a podvod k přinucení osoby, zasvěcené uvnitř cílené organizace, splnit požadavek, který obvykle zahrnuje zveřejnění informace nebo

vykonání akce, která napomůže útočníkovi. Může se jednat o prostý telefonát až po více komplexní situaci - přinucení oběti navštívit webovou stránku, která využije technické zranitelnosti a dovolí hackerovi převzít kontrolu nad počítačem."[9] Mitnick Kevin

Druh tohoto útoku se často označuje termínem "netechnický hacking". V oblasti informačních technologií se metody proslavili v 80. letech 20. století, kdy se o jejich popularizaci zasloužila mediální kauza kolem bývalého amerického hackera a sociálního inženýra Kevina Mitnicka. V dnešní době se jedná o jednu z nejrozšířenějších metod, jak z oběti získat požadované informace, či ji donutit udělat útočníkem zamýšlenou akci.

V roce 2011 zveřejnila firma Check Point® Software Technologies Ltd hlášení[27] z výzkumu "The Risk of Social Engineering on Information Security", ve kterém uvádí, že 48% z dotázaných společností se staly terčem útoku sociálního inženýrství 25 krát a více za uplynulé dva roky. Jeden úspěšný útok způsobil firmě finanční ztrátu 25 000$ až 100 000$, i více. Z hlášení vyplývá, že útoky byly zaměřené (seřazeno sestupně od nejvíce cíleného) na nové zaměstnance, dodavatele, výkonné ředitele, oddělení lidských zdrojů, vedení podniku a nakonec na informační oddělení.

Útočníci se snaží metodami sociálního inženýrství využít znalosti psychologie a faktory, jež ovlivňují chování a úsudek oběti, mezi které se řadí: důvěra, sympatie, nekonfliktnost, respekt, stres, soucit či jiné. Úspěšnost útoku závisí především na aktuální situaci, ve které se oběť nachází, na její povaze, pocitech, dále informovanosti, technických znalostech a různých jiných faktorech. Ve většině případů jsou rozhodující také schopnosti útočníka, zda dokáže v oběti vyvolat výše zmíněný pocit, či navodit požadovanou atmosféru. Dalším ovlivňujícím faktorem je vztah oběti k útočníkovi a délka známosti, či množství předchozí komunikace.

Sociální inženýr může využít různé druhy komunikace: osobní styk, telefonický hovor, zpráva, webová stránka, osobu třetí strany, atd. Ke každé z možností se může oběť stavět s jinou důvěrou a pozorností. Osobnímu styku či telefonátu bude oběť věnovat větší pozornost, než například elektronické zprávě. Elektronickou zprávou na druhou stranu může útočník oslovit velké množství adresátů za krátký čas. Sociální inženýr tedy musí vždy předem plánovat a vyhodnocovat své techniky.

Sociální inženýrství zavádí terminologii pro specifické metody a úkony:

- Phishing[8] - odeslání falšovaného e-mailu příjemci, který klamavým způsobem napodobuje legální instituci s úmyslem vyzvědět od příjemce důvěrné informace. Příjemce je většinou ve zprávě nabádán ke vstupu na webovou stránku přes přiložený odkaz. Ve scénáři phishingu je tato stránka vytvořena útočníkem a vzhledově napodobuje webové stránky instituce, za kterou se odesílatel vydává. Obsah zprávy či webové stránky vybízí oběť k zadání osobních údajů, přihlašovacích údajů k určité službě nebo k zadání čísla platební karty. Nerozpozná-li oběť, že se jedná o plagiát a vyplní-li důvěrné informace, jsou podvodnou stránkou uloženy, či odeslány útočníkovi. Tvůrce stránky může definovat, jaký další obsah bude oběti zobrazen. Může se jednat o

chybovou zprávu s následným přesměrováním na legitimní webové stránky společnosti, za kterou se ve zprávě vydává.

- Spam - nevyžádané poštovní sdělení šířené elektronickou zprávou. Většinou jsou zprávy zasílány hromadně, kdy cílí na mnoho příjemců. Adresy obětí jsou ve větším měřítku získávány různým způsobem - ukradeny z databází, automaticky vyhledány na internetu pomocí robotů, zaznamenávány přes podvodné stránky a další. Obsah zprávy pak obsahuje reklamní sdělení, phishingové sdělení a může také nést v příloze malware.
- Vishing - metoda, při níž je útok veden telefonickým hovorem namísto elektronickou poštou. Útočník se vydává za jinou osobu, nejčastěji zaměstnance legitimní společnosti, a snaží se z oběti vyzvědět potřebné informace.
- Pretexting - metody utváření vymyšleného scénáře útočníkem a jeho následné sdílení s obětí za účelem přinucení oběti vyzrazení informací či provedení akce. Falšována mohou být fakta, situace a především identita útočníka. Útočník často zakládá části informací na pravdě, aby navodil domněnku legitimnosti akce.
- Baiting - obrazně se dá přirovnat k útoku trojského koně v reálném světě. Pachatel zanechá na místě přístupném oběti zpravidla datový nosič či dokument označený lákavě vyhlížejícím popisem. Útočník doufá, že dříve či později se oběť pokusí spustit vnitřní obsah.

Z hlediska přístupu útočníka k oběti mohou nastat tyto scénáře:

- Přímý přístup - útočník se ptá přímým způsobem oběti na požadované informace.
- Důležitá osoba - útočník se identifikuje jako osoba z vedení firmy a většinou předstírá, že se nachází v situaci, kdy potřebuje rychle vyřešit určitý technický problém. Zdůrazní důležitost požadovaného činu či informací, na které se oběti zeptá. Pachatel v této situaci spoléhá na fakt, že dotyčná osoba uvěří identifikaci, bude jej respektovat jako nadřízeného a ráda pomůže.
- Bezmocná osoba - pachatel se identifikuje jako nový zaměstnanec či zaměstnanec stávající, který řeší problém se systémem. Cílem útoku většinou bývá technická podpora či správce systému. Útočník často předstírá jistou neznalost a doptává se na informace. V údernější variantě scénáře může předstírat, že řeší zapomenuté heslo či první pokus o přihlášení.
- Osoba technické podpory - pachatel se oběti snaží prokázat, že patří k technické podpoře nebo je on sám správcem systému. Uvěří-li oběť této lži, zpravidla pak sdělí požadované informace nebo vyplní útočníkem zadaný úkol.
- Reverzní sociální inženýrství - technika, kdy se útočník pokusí zaranžovat události

takovým způsobem, aby jej oběť sama kontaktovala s prosbou o pomoc. Obvykle je útok proveden ve třech krocích:

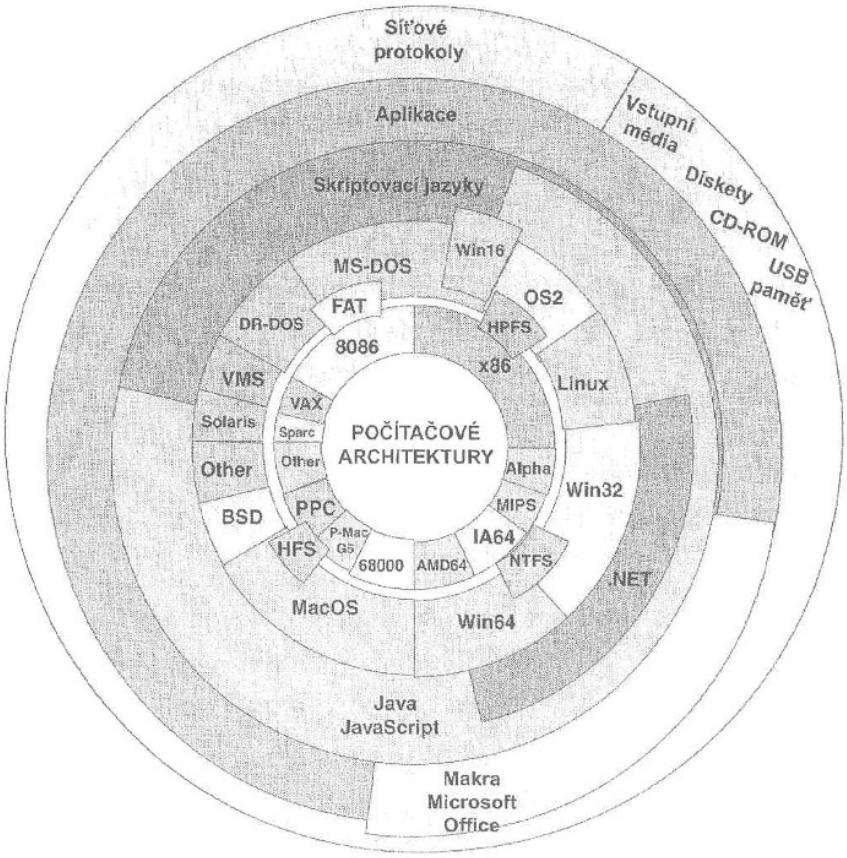
- Sabotáž - útočník zinscenuje nějaký problém a snaží se oběti na něj poukázat. Problém může být skutečný nebo fiktivní.
- Inzerce - pachatel nabídne technické řešení odstraňující problém či odbornou konzultaci a vyčkává, zda-li oběť zareaguje na jeho nabídku.
- Pomoc - pachatel problém odstraní, nebo prohlásí za odstraněný. Při vykonávání práce získává útočník informace od oběti a často si také vyžádá přístup do systému.

2.6 Cílový systém

V této fázi má pachatel potřebné, nebo alespoň směrodatné, informace o systému a službách běžících na cílovém stroji. Ze zjištěných informací si útočník udělá obraz o možnostech, které vzdálený systém nabízí. "Jedním z nejdůležitějších kroků pro pochopení počítačových virů je především dobré pochopení prostředí, ve kterém operují."[4] Každé prostředí nabízí jisté zdroje, komunikační prostředky a definuje závislosti, především pak formát proveditelného kódu. Programátor může využít ty možnosti, které mu prostředí poskytuje, ale současně je nucen dodržovat formáty a pravidla prostředím definovaná.

Existují mnohé závislosti, které ovlivňují vykonávání a funkce malware[4]:

- Závislost na počítačové architektuře
- Závislost na procesoru či jiném hardware
- Závislost na operačním systému a jeho verzi
- Závislost na souborovém systému
- Závislost na jiném homogenním subsystémů či programu
- Závislost na zranitelnosti
- Závislost na síťovém připojení



Obrázek 1: prostředí škodlivého kódu

Nepsaným pravidlem v oblasti vývoje software je jeho dělení do vrstev a abstrakce. Tím se řeší problémy s přenositelností kódu. I běhová prostředí se dělí dle vrstev, na kterých pracují:

2.6.1 Strojový kód

Ze software hlediska se jedná o nejnižší vrstvu, ve které může být program napsán - počítač dokáže zpracovávat jen určitý soubor instrukcí, které jsou definovány jeho procesorem. Instrukce jsou zapsány číselnými kódy a ovlivňují chování procesoru. Chce-li útočník vytvořit škodlivý kód pro tuto vrstvu, bude moci využít všechny možnosti procesoru, ale jeho kód bude limitován jen na kompatibilní procesory. Nebude-li řečeno jinak, bude se práce zabývat procesory architektury x86.

2.6.2 Operační systém

Existují různé operační systémy pro různé potřeby a pro různá zařízení. Práce bude směřována především na nejvíce používané OS - Microsoft Windows, MAC OS a Linux. OS obecně umožňují komfortnějšího ovládání počítače a práci s hardware. Odstraňují také přímé závislosti

uživatelských programů na konkrétním hardware - vytváří abstraktní vrstvu. Ve skutečnosti je OS ještě více rozvrstven.

Mezi nejdůležitější funkce systému patří: řízení přístupu k hardware, správa procesů a užití multitaskingu, správa paměti, práce se souborovými systémy, práce se sítí a jiné. V neposlední řadě zajišťuje také běh uživatelských programů.

Podaří-li se útočníkovi zavést svůj malware do systému, může využít všech možností, které mu OS nabízí. Často malware využívá API, které umožní využívat prostředky a funkce systému. Na druhou stranu každý z výše jmenovaných OS obsahuje (v aktuální verzi) bezpečnostní mechanismy, které limitují přístup aplikací přes rozhraní systému. Jedná se především o práva procesů. Každý nový proces se spouští pod určeným uživatelem, který má definovaná svá práva a přístup k souborům. Proces může při spuštění, i za běhu, požádat o jiná práva. Tato akce ale musí být potvrzena, a nebo musí být zadáno heslo požadovaného uživatele.

Při skenování služeb (viz kapitola "sběr informací") se pachateli podařilo získat informace o běžících službách. Jejich chod je zajišťován právě operačním systémem.

2.6.3 Běhová prostředí nad operačním systémem

Další vyšší vrstvou jsou běhová prostředí nad operačním systémem. Užívají určitý stupeň virtualizace a poskytují tak své vlastní funkce programům, které jimi budou spuštěny. Široká vrstva software využívá právě tuto vrstvu, včetně skriptovacích jazyků - dá se říci, že i webový prohlížeč interpretující JavaScript se řadí do této vrstvy, stejně tak kancelářská aplikace, jež umožňuje běh maker.

Tuto vrstvu vytváří i další produkty jako WINE, který zpracovává volání programů určených pro jiný systém a poskytuje vlastní knihovny. Dále pak interpreti bytekódu, kteří překládají bytekód na nativní - Common Language Runtime (.NET aplikace), Java Runtime Environment (Java aplikace) a další.

Je patrné, že různé závislosti způsobují nekompatibilitu mezi různými systémy. Na druhou stranu stačí, když malware splňuje požadavky jen toho systému, pod kterým je zamýšlen jeho chod. Příkladem buď virus implementovaný pro prostředí Oracle Java SE. Jelikož je možné prostředí Java Runtime Environment (JRE) provozovat nad operačními systémy (32 bitovými i 64 bitovými) Microsoft Windows, Apple Mac OS, Linux či jinými, bude možné spustit malware pod těmito systémy a nevzniká na této vrstvě přímá závislost na operační systém či architektuře. Vzniká ovšem závislost na subsystém JRE, který užívá a poskytuje zdroje nadřazeného systému. Pod různými druhy JRE tedy nemusí být vždy malware provozuschopný či přesně vykonávat své funkce. Dále vzniká závislost nepřímá, kdy funkce JRE jsou závislé na možnostech operačního systému.

Je potřeba zmínit, že vrstvení není vždy ve všech ohledech přesně definováno a abstrakce

nemusí být ve všech částech systému úplná. Systémem může být například dovoleno přepsání jiného paměťového místa, než určeného pro konkrétní proces, dále i přímí přístup k instrukcím procesoru a jiné. Je tedy vidět, že různé části vrstev se mohou prolínat.

2.7 Tvorba malware

Nyní se následné kroky pachatele rozdělí na dva různé scénáře:

1. Útočník se pokusí o nalezení vzdáleného místa náchylného na exploit. Podaří-li se, bude v návrhu a při programování již s touto možností počítat a přizpůsobí malware a jeho šíření tak, aby využíval nalezené zranitelnosti. Jinými slovy - nejdříve dochází k penetraci a následnému vytvoření malware.
2. Obrácený postup. Útočník se již předem rozhodl, že nejdříve vytvoří malware pro cílový systém a následně se jej bude pokoušet přenést a spustit. Tento postup je aplikován také v případě, kdy útočník nenalezne žádnou vzdáleně zneužitelnou slabinu a bude chtít penetraci provést jiným způsobem, než exploitem.

2.7.1 Programování

Před začátkem samotného programování si autor musí zvolit, pro jaký běhový systém bude aplikaci psát. Je výhodné mít také definovány akce, které po malware autor požaduje. Definované požadavky na malware mu ulehčí určit, jaký běhový systém bude nejvhodnější. Příklad: pachatel bude chtít využít známé zranitelnosti služby a implementovat ji do počítačového červa, který bude měnit registry systému a dále se šířit - proto je jasné, že nevyužije makro kancelářské aplikace, která úkol nezvládne.

Možná efektivita výsledného kódu bývá často přímo úměrná znalostem útočníka - jinými slovy, "čím více útočník cílové prostředí zná a dokáže využít nabízené možnosti, tím je schopen vytvořit sofistikovanější hrozbu". Je tedy nezbytné, aby si prostředí předem nastudoval. Cenným zdrojem informací jsou příručky k danému prostředí a publikace zabývající se zvoleným prostředím. Útočník si zpravidla dané prostředí nainstaluje na vlastní (i virtuální) stroj a současně s implementací svůj výtvor testuje.

Je běžné, že v průběhu studia, programování či testování nalezne útočník nová řešení, či jen změní názor na určité principy fungování malware. Závisí pak na jeho vůli, zda nové záměry aplikuje do produktu a nahradí tak část své práce.

K tomuto scénáři může dojít i v pozdější fázi, kdy je malware již aktivní na cílových stanicích. Vytvořil-li si autor přístup k napadeným stanicím pomocí svého malware, či existuje-li ještě cesta, jak stanici znova infikovat, může útočník provést aktualizaci své předchozí verze.

Otázka ohledně volby programovacího jazyka není příliš složitá - je ovšem potřeba si uvědomit, že programovací jazyk je pouhý nástroj, se kterým autor pracuje. Více než výborná znalost jazyka je potřeba ovládat umění algoritmizace a programátorských postupů. Mezi potřebné

postupy se řadí práce se soubory, s databázemi, se sítí a sokety, vícevláknové programování a další. Různé jazyky mohou přistupovat k těmto technikám trochu jiným způsobem, ale princip je prakticky stejný. Požadovaný programovací jazyk je v konečném důsledku závislý na kompilátoru či interpretačním software. Příkladem je Microsoft .NET Framework, pro který je možno využít spousty jazyků[28] (Visual C# .NET, Visual C++ .NET, Visual Basic .NET, JScript, Python pro Microsoft .NET a další), kompilátor pak zvolený jazyk převede na jednotný bytekód.

Časté řešení ovšem bývá v užití jazyka assembler pro programování malware na nejnižší vrstvě, v rovině operačního systému C, C++, assembler, či jejich kombinací, nebo využití dávkových souborů. Na vyšších vrstvách se pak často tvoří malware pomocí skriptovacích jazyků, jazyka python, C# a dalších.

2.7.1.1 Procesor a paměť

V následujících kapitolách budou probírány metody exploitace. Jelikož většina technik souvisí se změnou toku programu, která je způsobena narušením paměti, je potřeba se na paměť blíže zaměřit.

Aby mohl program vykonávat svoji činnost, potřebuje paměť k odkládání svých proměnných. Pamětí se nemyslí pouze operační paměť počítače, ale i procesor má svá vlastní paměťová místa, která se nazývají registry. Některá slouží jako pracovní registry, další odkazují na místa v paměti - především na zásobník, jiná uchovávají příznaky.

Každý proces získá přístup do paměti, která je rozdělena na segmenty[3][10]:

- Text - Slouží pro uchování kódu programu, velikost je pevná.
- Data - Segment ukládá globální proměnné programu, velikost je pevná.
- BSS - Uchovává neinicializované proměnné, velikost je pevná.
- Halda (anglicky "heap") - Část určená k dynamické alokaci proměnných. Jelikož kompilátor předem nezná velikost požadovaných proměnných, používá se k odkazování do této části ukazatelů. Stanou-li se dynamicky alokované proměnné nepotřebné, musí být uvolněny, aby nedocházelo k úniku paměti. Velikost haldy je tedy proměnlivá a roste od nejnižší adresy k nejvyšší.
- Zásobník (anglicky "stack") - Velikost je opět proměnlivá a plní se naproti haldě - od nejvyšší adresy po nejnižší. Na zásobník se ukládají lokální proměnné pro každou volanou funkci, ale především i návratová adresa. Každému bloku náležícímu k určité funkci se říká zásobníkový rámec.
- Prostředí - Paměťový segment ukládající parametry předané z běhového prostředí a kopie systémových proměnných.

2.7.2 Akce

K čemu by byl virus, který by nic nedělal? Útočník při programování vkládá do malware kódy, které budou na cílovém systému plnit jeho přání. Jak již bylo řečeno, možnosti mohou být takové, jaké dovoluje běhový systém.

Zde jsou uvedeny některé akce, které útočník po malware požaduje:

- Vytváření lokálních kopií
- Šíření svých kopií
- Aktivace po startu systému
- Ukrytí se v systému
- Vytvoření zadních vrátek
- Zahájení síťové komunikace
- Stažení dalšího malware
- Špionáž
- Editaci či mazání dat
- Provést útok
- Rozeslání spamu
- Mnoho jiného

Obsahově by bylo velmi rozsáhlé popisovat různá prostředí a v nich způsob provedení jednotlivých akcí. Práce proto bude zaměřena na systémy Microsoft Windows XP, Windows 7 a Windows 8. OS nabízí nepřeberné množství funkcí, které programátorovi zpřístupňuje pomocí API rozhraní - pod systémy Microsoft Windows se toto rozhraní označuje jako Windows API, Win32 API či WinAPI. Pomocí Windows API tedy může programátor realizovat činnosti jako práce se soubory, práce se sítí (WinSocket), sběr informací, ovládání hardware, přenastavení systémových registrů a mnohé další. Mezi důležitou součást patří také schopnost spustit jiné procesy.

Je-li pod operačním systémem spouštěn nový proces, obdrží práva uživatele, který jej vyvolal. Toto může být pro útočníka velmi limitující. Při volání důležitých funkcí jsou systémem kontrolována oprávnění a podle toho je akce povolena či zakázána. Dnešní souborové systémy také ukládají u souborů a složek značky, podle kterých systém hodnotí, zda k nim má proces přístup. Chce-li útočník zajistit, aby jeho malware mohl využít co nejvíce funkcí, bude potřeba získat co možná nejvyšší oprávnění.

Získání oprávnění:

- Spuštění privilegovaným uživatelem - má-li útočník štěstí, je jeho malware spuštěn pod uživatelským účtem s plným oprávněním. Program může být napsán způsobem, který jej registruje v systému a je spouštěn při každém startu. Stačí tedy, aby malware získal administrátorská práva pouze jednou a pak se registroval jako spouštěný s těmito právy.
- Znalost hesla - získá-li útočník heslo k privilegovanému účtu, může hesla využít ke zvýšení oprávnění.
- Využití chyb - je jasné, že při běžném provozu je spouštěno mnoho procesů s různým oprávněním. Nalezne-li útočník využitelnou chybu v procesu s vyšším oprávněním, může ji využít a získat práva napadené aplikace. Tomuto postupu se také říká "lokální exploitace".

2.7.3 Lokální exploit

Slovem exploit je označován specifický program (nebo sekvence dat), který využije chybu systému, či jiného programu, ke svému prospěchu. Cílem technik je převzetí kontroly nad tokem programu.[1]

Exploity mohou být děleny dle komunikace se zranitelným software na lokální a vzdálené. Lokální exploit se používá k získání privilegií aplikace, na kterou je exploit použit. Vzdálené exploity budou probírány později.

Většina exploitů souvisí s narušením paměti - v programu má každá alokovaná proměnná v paměti přidělené své místo a velikost. Pokusí-li se však proces uložit větší proměnnou, než pro kterou je paměťové místo vyhrazeno, přepíší se i data následující. Tato situace není kompilátorem ani systémem kontrolována, protože by to znamenalo výrazné zpomalení systému - integrita proměnné by musela být kontrolována při každém jejím ukládání do paměti.

2.7.3.1 Přetečení zásobníku

Každému spouštěnému procesu je přidělena část paměti obsahující segment typu zásobník. Jak již bylo zmíněno, zavoláním funkce z programu se vytvoří na zásobníku nový zásobníkový rámec, který obsahuje také návratovou adresu z funkce. Cílem útoku je provést přetečení do této návratové adresy a nastavit hodnoty na útočníkem definované paměťové místo. Pachatel také může do paměti vložit vlastní instrukce a přinutit program udělat zamýšlenou akci. Těmto vloženým instrukcím se říká shellkód.

2.7.3.2 Přetečení na haldě

Paměťový segment halda neslouží k řízení toku programu, ale lze jej taktéž využít k útoku. Na haldu se ukládají dynamicky alokované proměnné, které mohou nést důležitá data. Například cestu k souboru či síťovou adresu. Podaří-li se nalézt místo v programu, které je zranitelné na přetečení, může útočník na následující paměťové pozice zapsat data, která požaduje. Tímto

způsobem kompromituje například adresu souboru, do kterého se budou ukládat důležitá data nebo síťovou adresu na kterou program bude vytvářet připojení.

2.7.3.3 Přetečení v segmentu bss

Do tohoto paměťového segmentu se ukládají globální proměnné. Globální proměnnou může být i ukazatel na funkci. Podaří-li se útočníkovi kompromitovat hodnotu proměnné, může ji pozměnit na jinou, požadovanou funkci. Ukazatel tak odkáže vykonávání programu do jiné funkce.

Existují i různé další taktiky, jak procesu podvrhnout data - programátor si však musí nastudovat všechny vstupní cesty a podle toho nejlépe zvolit využitelnou metodu. Hledání nových chyb je často běh na dlouho trať, jelikož ne vždy musí program konkrétní chyby obsahovat a ne vždy lze nalezené chyby využít. Hacker tak tráví mnoho času nad ladícími programy a kódem programu, který chce napadnout.

Ne vždy potřebuje útočník nalézt v programu úplně novou chybu, ale vystačí si již s chybou, kterou objevil někdo jiný. Existují veřejné databáze zneužitelných chyb, které útočníkovi ušetří mnoho času. Tyto chyby bývají většinou zveřejněny legálním způsobem a to až po určitém čase, kdy autor postiženého programu již vydal opravenou verzi svého produktu a uběhla dostatečně dlouhá doba potřebná k aktualizaci na klientských stanicích.

O těchto databázích, a jejich použití, bude řeč v kapitole věnované penetračnímu testování.

2.7.4 Ukrytí

Je-li útočníkovým záměrem svůj kód v cílovém prostředí ukrýt, bude vyžadováno, aby byl co nejméně nápadný pro uživatele a bezpečnostní programy. Opět platí, že možností ukrytí je mnoho a záleží na konkrétním typu malware a kreativitě útočníka.

2.7.4.1 Ukrytí před uživatelem

Je patrné, že míra odhalení také často závisí na znalostech uživatele. Kód se v systému bude projevovat svými akcemi a také bude někde přechováván.

- Aktivace kódu po spuštění systému - Velká část malware se registruje do systému tak, aby byl jeho kód spuštěn po načtení systému či přihlášení uživatele. OS Microsoft nabízí několik cest, jak toho dosáhnout. Vložení kódu či odkazu do složky "po spuštění", vložení záznamu do registru, aktivace pomocí plánovače, nebo registrace kódu jako systémové služby. Útočník může využít i sofistikovanějších metod - připojit malware na některý ze spouštěných procesů, tak bude spuštěn současně s tímto procesem. Nemusí se vždy jednat o systémovou službu, připojit se může i k programu, který bude uživatelem patrně často spouštěn ručně.

- Uložení - většina malware je naprogramována způsobem, kdy uloží svůj kód na datový nosič systému. Pachatel většinou nevolí místo uložení tak, aby se kód nacházel na zřetelném místě. Často je ukryt v adresářové struktuře mimo uživatelem běžně používané složky.
  I zde existují pokročilejší metody, jak kód skrýt. Může se jednat o připojení kódu malware na jiný spustitelný soubor či umístění do specifické oblasti na disku - příkladem může být Alternate Data Streams, který je podporován souborovým systémem NTFS. Implementován byl z důvodů kompatibility se souborovým systémem HFS, současně však vytvořil vhodné místo i pro malware. Běžně se soubory ukládají do primárního datového proudu, útočník ovšem použije přesměrování toku a nový soubor pojmenuje ve formátu "původní:nový". Tím zůstane původní soubor na svém místě v primárním proudu, ale v ADS se vytvoří soubor "nový". Prochází-li uživatel své soubory, vidí právě ty, které jsou v proudu primárním, soubory v ADS ale již nevidí. Dokonce se nezobrazí ani jiná velikost původního souboru.
- Správce procesů - Správce procesů OS Windows zobrazuje aktivní procesy všech uživatelů. Existují možnosti, jak nově vytvořený proces skrýt. Mezi nejvíce používané patří registrovat proces jako službu, či připojit program na jiný. Dokonce i v případě, že je aplikace spuštěna z ADS, je ve správci viditelná jako proces z primárního proudu. Pachatel může uživatele také oklamat tím, že dá programu jméno vyhlížející jako systémový proces nebo důvěryhodná aplikace.
- Akce - uživatel může rozpoznat malware podle jeho projevů. Situace, kdy malware smaže uživateli data, jej rychle prozradí. Existují však i jiné, méně nápadné, projevy - zvýšená aktivita procesoru, delší odezva při spuštění aplikace, neočekávané pády, zvýšená a nepovolená síťová komunikace, otevřený port pro příchozí spojení.

2.7.4.2 Ukrytí před bezpečnostním software

Právě antivirové programy jsou ty, které chrání uživatele před usídlením a spuštěním nákazy. Cílem útočníka je tento software vyřadit z provozu, nebo před ním svůj malware ukrýt - mezi nejvíce užívané techniky patří matení kódů a kódování.

Podle maskovací techniky se viry dělí do skupin:

- Kódované viry - Tvůrce viru využije algoritmu, kterým zakóduje tělo viru a do jeho vstupní funkce vloží dekódovací algoritmus ("dekryptor"). Dekryptor musí být stále stejný, aby mohl tělo viru dekódovat. Často je využito i klíče, který každou další kopii viru dělá jedinečnou[6].
- Oligomorfní viry - Vylepšuje kódované viry tím, že ve svých kopiích mění i svůj dekryptor a každá následující generace je odlišná nejen svým tělem, ale i dekryptorem.
- Polymorfní viry - Pokročilejší oligomorfní viry - přidávají nadbytečné instrukce, různé

formy kódování, vícevrstvé kódování, v neposlední řadě také prohazují pořadí instrukcí, ale způsobem, aby výsledná akce byla ekvivalentní. Ve výsledku mohou vznikat až milióny různých exemplářů.

- Metamorfní viry - Neobsahují dekryptor, ani nemají konstantní tělo a jsou schopné vytvářet odlišné kopie[4]. Nevytváří žádná konstantní data či struktury, pomoci kterých by byly odhaleny. Metamorfní viry mohou být například distribuovány ve formě překladače a zdrojového kódu, který je zakódovaný. Při kompilaci se současně dekóduje i zdrojový kód, jsou přidávány instrukce, mění se jejich pořadí a zdrojový kód je znova kódován, ale jiným klíčem.
- Retro viry - Jejich úkolem je zaútočit na bezpečnostní software a pokusit se jej vyřadit či modifikovat tak, aby neblokoval následné akce. Tím připraví cestu pro další malware.

2.7.5 Šíření

Autor malware často požaduje, aby bylo zajištění jeho šíření. Malware je většinou naprogramován tak, aby po spuštění vytvořil svou kopii na lokálním systému, která je znova spouštěna při každém zapnutí stroje. Často se malware snaží šířit i na další vzdálené systémy skrze přenosná média či síť.

2.7.5.1 Šíření na lokálním systému

- Kopírovací metoda - malware má definováno, že bude vytvářet své nové kopie na disku počítače a také na výměnná média při jejich připojení.
- Přepisovací metoda - malware prochází adresáře a hledá soubory zvoleného typu, ty přepíše svým tělem. Původní soubor je tedy trvale poškozen.
- Parazitická metoda [12] - Na rozdíl od předešlé metody soubor nepoškodí, ale připojí k němu svůj kód. Malware se může připojit před kód původního programu, za program, ale také vložit mezi části původního programu. Vir pozmění informace v PE hlavičce, aby zajistil svou aktivaci.

2.7.5.2 Šíření na vzdálené systémy

Vyměnitelná média - Stejný princip jako v předešlé části. Malware zkopíruje kód na vyměnitelné médium.

- Síťová komunikace - Mnoho služeb umožňuje síťovou komunikaci. Pro útočníka je tato metoda výhodná, jelikož k rozesílání může zpravidla využít účet a kontakty oběti. Příjemce pak obdrží zprávu, jehož adresátem je napadený uživatel. Malware pak často mění název souboru.
- Elektronická pošta - Často používaný nástroj. Malware prohledá adresáře poštovních

programů a na nalezené adresy se rozešle - nejčastěji jako příloha, či formou odkazu.

- Služba instantních zpráv - Různé služby nabízí možnost zasílání zpráv v reálném čase. I tyto služby jsou napadnutelné. Malware se rozešle na kontakty ve formě souboru či odkazu na webové stránky.
- Sdílení adresářů - Malware může přenést svůj kód do vzdálených adresářů, či sdílet sám sebe.
- Jiné - Malware může využít i jiné služby ke svému přenesení. V neposlední řadě může nést i vzdálený exploit, který zneužije bezpečnostní mezery.

Možností šíření se nabízí celá řada, ne všechny cesty jsou proto jmenovány.

2.7.6 Komunikace

V tomto bodě má útočník provozuschopný malware, který po napadení cílového systému vykoná předdefinované akce. Ale to je vše. Chce-li mít autor možnost ovládat svůj program i po té, co je již aktivní na cílovém stroji, musí do něj implementovat mechanismy, které to povolí. Je jasné, že cílový stroj pak musí být dosažitelný skrz internet (či vnitřní síť).

Komunikace v internetu je založena na modelu TCP/IP, proto je potřeba implementovat technologii, která tuto komunikaci umožní. Na úrovni OS Windows ji implementuje rozhraní Winsock. Socket definuje koncový bod komunikace - aplikace vytváří sockety k zahájení odchozího spojení a také k naslouchání na spojení příchozí[7].

Komunikace se dělí dle toho, kdo spojení navazuje:

- Útočník - malware je implementován způsobem, kdy naslouchá na zvoleném portu. Útočník se vzdáleně na předvolený port připojí a může začít komunikace. Komunikace je obousměrná,takže zdroj i cíl mohou současně na sokety zapisovat i číst. Je patrné, že přenos sice funguje a malware bude zprávy číst, ale to je vše. Proto musí být implementován komunikační protokol, kterému budou rozumět obě strany. Pak již bude umět malware rozpoznat požadavek na akci, splnit jej a zaslat útočníkovi odpověď. Toto řešení má dvě velké nevýhody - je-li cílová stanice za firewallem, bude sice naslouchat, ale požadavek na spojení bude firewallem zakázán, dále je nutné, aby měla stanice stále stejnou IP adresu, či platný záznam v DNS službě. Proto je toto řešení uplatnitelné pouze u serverů, které mají nastavené směrování na pevnou veřejnou IP adresu.
- Oběť - v tomto případě útočník zpřístupní server, který naslouchá na své adrese a definovaném portu. Do těla malware je pak tato adresa (či doménové jméno) vložena. Malware může být nastaven způsobem, že se připojí k serveru a udržuje spojení aktivní, a nebo je spojení navazováno jen v určitých časových intervalech a malware si stáhne novou definici příkazů či svou aktualizaci. Útočník může chránit svou identitu tak, že

při každé aktualizaci se změní také adresa na které proběhne následná aktualizace. To ale přináší práci s neustálým přesouváním serveru. Jednodušší způsob je využít k přístupu na server anonymizační proxy ze strany pachatele - proxy serverem může být i jiný napadený stroj.

2.8 Penetrace

Cíl je definován - přenést malware na stroj oběti a aktivovat. Přichází čas, kdy budou ověřeny schopnosti a kreativita útočníka. Penetrace, nebo-li vniknutí cizího kódu, je vrcholnou, a většinou také nejtěžší, částí celého procesu. Je běžné strávit většinu času při tvorbě malware právě nad studiem cílového systému a hledáním efektivních cest k přenesení a spuštění hrozby.

Existuje sice množství cest, ale každá je jinak nápadná pro uživatele, časově náročná, přístupná, ale také ne každá z nich zaručí výsledek. Možností také je, že sebelepší malware skončí právě u tohoto kroku. Budou popsány různé cesty, které lze při šíření malware využít:

2.8.1 Fyzicky přístupný stroj

Pro útočníka je tato situace nejvíce přívětivá, ale také často nedostupná. Obzvláště, je-li útočníkovým cílem širší skupina strojů. Útok na lokální stroj není příliš složitý - existuje více metod, ale zpravidla se využívá odděleného systému na datovém nosiči útočníka. Ten se zavede namísto původního systému a následně může provádět prakticky jakékoli úpravy na discích lokálního počítače, včetně zkopírování souborů obsahující hesla (v hash podobě). Obrana proti takovému útoku nespočívá ve vytvoření hesla pro BIOS a nastavení priorit zavádění - útočník může lehce nastavení BIOS obnovit. Obrana spočívá v zašifrování disků a také především v zamezení onoho fyzického přístupu neoprávněným osobám. Je možné, že útočník také zkopíruje celý obraz zašifrovaného disku - bude-li ovšem užito bezpečného šifrovacího algoritmu a silného hesla, neměl by být útočník schopen heslo dekódovat v rozumném čase.

2.8.2 Vzdálený exploit

Chce-li být útočník co možná nejméně nápadný, zvolí jako další postup testování konkrétních služeb, jsou-li náchylné k exploitaci a následnému rozšíření viru do systému.

V minulých kapitolách bylo napsáno, že existují různé databáze uchovávající exploity k již nalezeným zranitelnostem[31][32] a to jak lokálním, tak i vzdáleným. Chce-li pachatel vzdálený exploit provést, ale současně i strávit co možná nejméně času nad hledáním zranitelností, je to pravé místo, kde začít. K dispozici je také opensource produkt "Metasploit", jehož cílem je usnadnit vyhledávání zranitelných míst a použití exploitů.[4] Metasploit obsahuje svou vlastní databázi exploitů a nálož (anglicky "payload"). Nálož je kód, který bude po úspěšné penetraci spuštěn.

Útočník si vytvoří konfiguraci počítače oběti ve svém vlastním prostředí a nainstaluje

požadované služby tak, aby se verze shodovaly s cílovým strojem. Nyní může začít testovat zranitelnost - ve vnitřní databázi produktu metasploit vyhledá exploity k testované službě a vyzkouší, zda je na ně služba náchylná. V textu již bylo zmíněno, že zranitelnosti se do databází dostávají až po delším čase, proto je možné, že služba (v testované verzi) nebude na žádnou z těchto zranitelností náchylná. Pachatel pak může využít ostatních databází - je možné, že exploit z databáze bude potřeba upravit do podoby, aby jej Metasploit dokázal interpretovat. I zde může nastat situace, že exploit nemusí být aplikovatelný.

Útočník se tedy rozhodne vytvořit exploit vlastní. Cenným zdrojem informací pro něj mohou být přímo webové stránky produktu, na který se rozhodne zaútočit. Autoři programů často uvádí seznam oprav, které provedli v nové verzi a to včetně těch bezpečnostních. Je-li tedy vzdálená služba neaktuální, může se útočník pokusit vytvořit exploit na tuto zranitelnost a napadnout tak cílový stroj.

Pachatel se může pokusit najít zranitelnost vlastními silami. Je potřeba si uvědomit, že oproti lokální službě je nyní komunikace omezena pouze na spojení přes sockety. I tak může systém obsahovat chyby, které často souvisí s formátováním řetězce či přetečením paměti. Mezi nejznámější chyby způsobené neošetřením vstupu patří Cross-site scripting a SQL injection - ty sice neumožní spuštění binárního kódu, ale mohou útočníkovi pomoci při získání cenných informací. Obzvláště, dostane-li se pomocí SQL injection k tabulce s hesly.

Je-li služba náchylná na útok přetečení paměti, může tvůrce exploitu využít techniky popsané v kapitole "lokální exploit" ovšem s rozdílem, že využije shellkód tak, aby zprostředkoval stažení malware.

2.8.3 Získání přihlašovacích údajů

V případě, že na cílovém stroji běží služba, která umožňuje vzdálené přihlášení, může se k ní útočník pokusit získat přihlašovací údaje. Většinou je vzdálená správa povolena na serverech, na osobních počítačích méně. Pachatel se ovšem může pokusit změnit (pomocí vzdálené správy) konfiguraci na směrovačích a přepsat adresy DNS serverů. Odešle-li oběť dotaz na útočníkův DNS server, bude mu zaslána jiná IP adresa. Při prohlížení webových stránek to bude mít za následek načtení jiné stránky. Ta může obsahovat škodlivý kód zneužívající chyby prohlížeče nebo nabídnout malware ke stažení.

K získání přihlašovacích údajů může vést mnoho cest. Útočník se může pokusit o uhodnutí hesel slovníkovým útokem, či útokem hrubé síly. Vychází-li útočník z předpokladu, že uživatel používá stejné přihlašovací údaje i na jiných službách, může se pokusit také o jejich získání.

2.8.4 Metody sociálního inženýrství

Sociálnímu inženýrství byla věnována celá kapitola, proto zde budou rozebírány pouze postupy související s penetrací malware. Do této metody spadají techniky, které přimějí uživatele, aby malware sám spustil a nebo dopomohl k jeho spuštění. Využívá se především maskování -

malware se může vložit do jiného programu, či se připojit na spustitelný soubor, jak bylo popisováno výše. Často autoři pouze mění název souboru a před koncovku vloží další příponu jiného formátu. Útočník pak takovýto soubor odešle elektronickou poštou, předá na výměnném médiu, či vystaví na stránky, kam oběť přiláká.

2.9 Životní cyklus

I malware má svůj životní cyklus. Může být rozdělen na jednotlivé fáze: vypuštění, rozšíření, aktivita, odhalení, úpadek, zánik. Dělení ovšem není vždy aplikovatelné - příklad může být malware, který se nepodaří rozšířit, malware který provede svůj update a znova musí dojít k jeho odhalení, či malware který je dosud neobjeven a stagnuje.

Pro útočníka není vždy lehké udržet svůj malware v oběhu. Je-li detekována nová hrozba, kterou antivirové programy dosud nerozeznávají, výrobci přidají její definici do databáze svých antivirových programů. Uživatelé si nové definice stáhnou a antivir již bude moci malware odhalit. Podobná situace může nastat i u malware, jež využívá určité chyby v systému. Autor implementuje záplatu a vydá novou verzi programu. Po stažení nové verze již není možné tuto mezeru využít a malware přejde do fáze úpadku či stagnace. Chce-li být pachatel úspěšný, musí hledat nové způsoby, jak vylepšit svůj malware, či provádět své akce velmi nepozorovaně. Je-li totiž na stroji oběti chyba opravena a malware eliminován, ztrácí útočník nad takovýmto strojem kontrolu.

3 Praktická část

Jedním z cílů projektu je praktická tvorba testovacího malware. V předešlé kapitole byly definovány teoretické postupy, jak cíle dosáhnout. V této části budou popsány praktické kroky, které byly autorem učiněny při tvorbě testovacího programu.

3.1 Virové generátory

Často označovány jako "Virus Generator Kits". Má-li pachatel v úmyslu vytvořit počítačový vir, nemusí vždy nutně vytvářet nový exemplář. Existují generátory virů, které dokáží vytvářet škodlivé struktury. Útočník si navolí požadavky z předdefinovaných v generátoru, ten pak sestaví virus. Tato činnost dokonce nevyžaduje programátorské schopnosti a je realizovatelná prakticky kýmkoli, kdo nebude mít problém s ovládání prostředí generátoru. Otázkou ovšem zůstává, jak odhalitelný a použitelný bude výsledný kód.

3.1.1 Virus Construction Set (1990)

Byl prvním programem určeným ke generování virů. Generátor se otáže na název textového souboru, který má být zahrnut a na počet souborů, které má infikovat. Vytvoří soubor VIRUS.COM, který po spuštění vyhledává další soubory s příponou "com" a infikuje je. Po dosažení celkového počtu předdefinovaných infekcí přepíše soubory config.sys a autoexec.bat a vypíše text z textového souboru.

Výsledný kód je jednoduchý, stále stejný a lehce odhalitelný.

3.1.2 VBS Worm Generator (2000)

Generátor obsluhovaný přes grafické rozhraní, který dle předvolených požadavků vytvoří Visual Basic skript. Předdefinované akce mohou být následující: vložit klíč ke svému startu do registru, zkopírovat kód na jiná místa v systému, odeslat svou kopii pomocí aplikace Outlook, přepsání specifikovaných typů souborů a jiné. Výsledný červ se také může spustit jako paměťově rezidentní.

3.1.3 Senna Spy Internet Worm Generator (2000)

Další z generátorů, jehož výsledkem je Visual Basic skript spustitelný pod systémy Microsoft Windows. Kód může rozesílat sám sebe skrze aplikaci Outlook, využít síťové komunikace a šifrování kódu.

3.1.4 NGVCK (2001)

Zástupce generátorů, které již nevytváří "pouze" Visual Basic skript, ale spustitelný soubor. Je

založen na metamorfním engine. Výsledný malware napadá PE hlavičky spustitelných souborů, obsahuje obranu proti lazení přes SoftIce Breakpoint, v neposlední řadě také šifrování ROR/ROL, NEG, NOT, XOR a ADD/SUB.

3.1.5 Novější generátory

Mezi zástupce se řadí DELmE's Batch Virus Generator, In Shadow Batch Virus Generator, JPS Virus Maker či TeraBIT Virus Maker. Jak již názvy napovídají, jedná se o generátory dávkových souborů. Možnosti dávkových souborů jsou široké a jejich programování poměrně jednoduché. Představují tedy hrozbu především pro neinformované uživatele, kteří výsledné soubory spustí.

Pokusí-li se útočník na internetu vyhledat virové generátory, bude zahlcen právě takovými, které generují dávkové soubory nebo využívají skriptovacího jazyka. Pro programátory většinou nebývá přílišně těžké tyto generátory vytvořit, stejně jako skripty samotné. Proto není divu, že jich je, na rozdíl od těch sofistikovanějších, mnoho.

Pro zkušeného útočníka zřejmě tyto generátory nebudou příliš zajímavé, jelikož ten bude chtít využít sofistikovanějších metod a přizpůsobit chování malware přesně podle svého záměru.

3.2 Cíle praktické části

Úkolem je vytvořit takový malware, který by svou charakteristikou splňoval požadavky na škodlivý kód. Tento malware bude provozuschopný, ale testován bude pouze v prostředí autora. Cílem projektu je v průběhu tvorby testovat míru obtížnosti, jakou je třeba zvládnout při tvorbě malware, ověřit definované postupy a zjistit, jaké znalosti a schopnosti musí případný pachatel ovládat, aby jeho práce mohla být úspěšná.

Dále budou testována bezpečnostní opatření systémů a odolnost oproti implementovanému malware. V neposlední řadě bude diskutováno, jak zvýšit odolnost svých systémů vůči počítačovým hrozbám.

3.2.1 Požadavky na malware

- Malware bude přeložen do spustitelného PE formátu pro systémy Microsoft Windows XP, 7 a 8
- Po spuštění se aktivuje na systému a bude spouštěn při každém startu
- Jeho chod bude možné vzdáleně řídit a zasílat řídící příkazy, které malware vyplní

3.2.2 Kritéria a hodnocení

- Penetrace nebude z právního hlediska prováděna na žádný jiný stroj, budou ovšem

diskutovány a zhodnoceny možnosti šíření

- Bude provedeno testování, zda-li je malware funkční dle požadavků a svým charakterem odpovídá škodlivému kódu
- Budou diskutovány možné škody způsobené rozšířením
- Budou doporučeny postupy k prevenci před nákazou a možnosti detekce

3.3 Příprava

V tomto projektu se autor drží postupů, které jsou specifikovány v teoretické části tohoto dokumentu a to od kapitoly "Důležitá příprava", jelikož právě toto je vstupní bod při začátku nového projektu.

Jsou zde popsány požadavky, které si autor musí definovat ještě před začátkem projektu. Většina z nich je již definována samotným zadáním této práce, ale nyní je čas rozšířit zadání na konkrétní výslednou podobu projektu.

3.4 Struktura malware

Aby malware odrážel některý ze skutečných scénářů, je jeho vývoj směřován za účelem vytvoření (fiktivní) botnet sítě - tedy sítě napadených počítačů, kterou je možno vzdáleně ovládat. Botnet je v praxi používán především k útokům zvaných odmítnutí služby (anglicky Denial of Service). Po vytvoření sítě je útočníkem zaslán příkaz na dostupné počítače, který obsahuje informace o cílí, na který mají stroje obětí zasílat požadavky. Cíl, na který je DoS útok směřován, obvykle nevydrží reagovat na stále nově příchozí požadavky a dojde k zahlcení. V tomto bodě již server není schopen přijímat připojení od jiných klientů, kteří chtějí regulérně využít jeho služby. Hackeři často pronajímají služby botnet sítě na černém trhu.

Je vidět, že k vytvoření takovéto sítě je třeba architektury typu server-klient. Proto jsou v rámci projektu vytvořeny dvě aplikace - serverová část a klientská (malware).

Sběr informací o vzdáleném systému, popsaný v teoretické části, zde není uvažován, jelikož malware směřuje na obecný cíl a jeho snahou je infikovat vícero uživatelů (jak již bylo řečeno, nejedná se o reálné uživatele, jelikož by se stal postup protizákonným). Malware je implementován pro systémy Microsoft Windows, jelikož jde o nejrozšířenější systém na uživatelských stanicích[33] (březen 2014). V oblasti serverů je nejvíce rozšířeným systém Linux, serverová část je proto (a pro rozmanitost projektu) implementována pro tuto platformu.

Po vymezení podoby malware je třeba vytvořit prostředí, ve kterém probíhá implementace a testování. Jako hlavní systém je zvolen Linux Debian, v němž jsou instalovány virtuální stroje pro běh systémů Microsoft Windows XP, Microsoft Windows 7 a Microsoft Windows 8. Při implementaci může nastat problém díky některým rozdílnostem novějších a starších systémů, na které malware cílí. Především pak v práci se specifickými funkcemi a programy. Proto se

doporučuje postupy testovat v každém ze systémů a řešit případné odlišnosti.

3.5 Cílové operační systémy

Nyní je zapotřebí nastudovat možnosti, které nabízí cílové běhové prostředí pro malware, ale i pro server. To zabere poměrně mnoho času, jelikož možnosti jsou velké. Microsoft Windows poskytuje programátorům WinAPI, které zprostředkovává množství funkcí systému. Malware proto užívá těchto funkcí.

Serverová část běží na jiném stroji a simuluje prostředí útočníka. Prostředí založené na Linuxu nabízí programátorovi téměř neomezené funkce pomocí knihoven.

3.6 Programování

Cílem je, aby byl malware spustitelný v systému bez dalších závislostí a mohl využít možností, které systém nabízí. K tomuto účelu nejlépe slouží formát executable (exe). Výsledný formát lze ovlivnit nastavením kompilátoru, musí se ovšem zvolit takový kompilátor a jazyk, ze kterého bude možno formát exe vytvořit.

Mezi jazyky, které nejlépe vyhovují požadavkům patří assembler, C a C++. Jazyky je možné poměrně snadno kombinovat v rámci jednoho projektu, což přináší značnou výhodu. Jazyk assembler je též označován jako "jazyk symbolických adres" a je užíván k programování na nejnižší úrovni. Jeho užití je vhodné zejména proto, že dává programátorovi kontrolu nad každou vykonávanou instrukcí. Je ovšem zbytečné jim řešit různé algoritmické struktury, které se dají efektivně vyřešit vyšším jazykem. Programovací jazyk C se řadí do skupiny vyšších programovacích jazyků, ale i přesto dává dostatečnou kontrolu nad prací s pamětí. C++ přináší hlavní výhodu v rozšíření jazyka C o objektově orientované programování.

To vše se týkalo vývoje malware. Serverová část je programována pro Linux a opět je požadováno, aby její běh zprostředkoval přímo systém. Je tedy užito jazyka C/C++ a systémových knihoven.

V závislosti na jazyku si autor vybírá vývojové prostředí. Lze použít obyčejný textový editor, ale pro lepší přehlednost se často volí některé z prostředí, které je svými funkcemi specializováno na vývoj v určitém jazyku. Jako vhodný kandidát se jeví opensource software Code::Blocks, který je dostupný jak pro platformu Linux, tak Windows. V neposlední řadě zbývá zvolit kompilátor. Standardním nástrojem pod systémy Linux je g++. MinGW podporuje překlad zdrojového kódu do binární podoby pro systémy Windows. Kompilátor MinGW je možné použít pod systémy Linux i Windows. Proto je zvolen i v tomto projektu. Nadešel čas na programování.

3.6.1 Implementace malware

Kapitola popisuje jednotlivé části, postupy a funkce malware tak, jak jsou implementovány

autorem. Dále zmiňuje technologie, které byly při návrhu použity, a rozvádí jejich principy. Podkapitoly jsou řazeny způsobem, aby odrážely postupné vykonávání kódu.

3.6.1.1 Po spuštění

Při prvním spuštění malware je požadováno, aby vykonal mnoho akcí - skryl své provádění, zkopíroval sám sebe, registroval do systému svá další spuštění a také umožnil komunikaci.

Ze všeho nejdříve je potřeba zamaskovat malware tak, aby při spuštění nevykazoval žádné známky chování, které by zaregistroval uživatel - především pak zobrazení okna. Způsob ukrytí závisí na povaze aplikace, zda-li se jedná o aplikaci konzolovou či grafickou. V obou případech nabízí WinAPI funkce, které skryjí konzoli, či okno aplikace.

Je potřeba také vyřešit určité rozdílnosti v různých verzích systému Windows. Některé funkce a programy mohou být nepoužitelné v jedné z verzí, některé se zase mohou chovat odlišně. Proto jsou do malware implementovány postupy, které zjistí verzi operačního systému a uloží zjištěnou hodnotu. Hodnota je využita v situacích, kdy je další vykonávání závislé na aktuální verzi systému a podle tohoto se program dále větví.

Nyní přichází část, kdy malware bude vytvářet kopii sama sebe, aby se zajistilo jeho uchování na hostitelském stroji. Nejprve je potřeba zvolit místo zápisu a pojmenování souboru. Z důvodů utajení užívá malware jako název souboru stejné jméno, které pojmenovává jeden ze systémových procesů - "svchost.exe". Cesta k souboru je zvolena tak, aby byla pro uživatele také co nejméně nápadná. Malware se ukládá do uživatelské složky pro místní nastavení systému Windows. Tím je zajištěno, že uložení je povoleno i v případě, kdy je soubor spuštěn uživatelem nevlastnící administrátorská práva.

Následuje registrace zkopírovaného programu do systému, aby jej systém při každém svém zavedení spustil. Existují různé cesty, jak tohoto dosáhnout. V projektu přidává malware svůj klíč do uživatelských registrů. Při každém spuštění se ověřuje, zda je hodnota stále v registru přítomna. Není-li, malware ji opět vloží. Stejně tak se ověřuje i přítomnost souboru, který se v případě potřeby znova zkopíruje.

Proces pokračuje a pokouší se nyní vložit záznam do Windows firewall a umožnit tak komunikaci se serverem. Malware vkládá pravidlo, které povoluje komunikaci na portu 1025 protokolu TCP. Název vkládaného pravidla se maskuje pod názvem "Windows Updates". Je třeba říci, že právě tato sekce může být kritická, jelikož k vložení pravidla do firewallu je zapotřebí administrátorského oprávnění. Není-li odchozí komunikace bránou firewall kontrolována, nejedná se o problém a spojení se naváže. V opačném případě je potřeba zvolit takovou strategii, která problémy s bránou firewall eliminuje co nejvíce. Strategie je cílena přímo na uživatele pomocí sociálního inženýrství. Nepodaří-li se vložit pravidlo do systému, malware se pokusí přímo o komunikaci - aktivní firewall ji zachytí a uživateli se ukáže výzva ke schválení. Jelikož se malware svým jménem vydává za systémový proces, je možné, že uživatel spojení odsouhlasí a pravidlo uloží. V případě, že uživatel nebude proces znát, je

pravděpodobné, že se jej pokusí nalézt na internetu pod jménem "svchost". Z dokumentace se dočte, že se jedná o systémový proces a odsouhlasí jeho komunikaci.

3.6.1.2 Zvýšení oprávnění

K vývoji malware je často požadováno i kreativní myšlení. Útočník se často zamýšlí nad požadovaným cílem a možnostmi jeho dosažení. V reálném projektu se pachatel soustředí také na způsoby zvýšení oprávnění pro jeho program. V rámci projektu byla zvolena možná varianta taková, při niž bude užit útok hrubou silou na funkce umožnující zvýšení oprávnění a spustit tak proces za pomoci uživatelského jména a hesla. Funkce se nazývá "LogonUser". Uživatelská jména je možno získat přes WinAPI. Hesla samozřejmě viditelná nejsou, není k nim umožněn ani přístup. I v případě získání těchto hesel jsou uložena ve formě hash otisku a není možno je přímo aplikovat ve funkci, která oprávnění zvýší.

Nabízí se možnost provedení útoku hrubou silou. Tedy takového, kdy budou postupně generovány všechny možné variace s opakování a aplikovány na funkci. Využití je ovšem diskutabilní z prostého důvodu: ve funkci je jejími tvůrci záměrně aplikována prodleva, která zpomalení proces získávání hesla. Proto tento postup není přímo aplikován ani v malware, ale zdrojový kódy autor implementoval samostatně (a je součástí přílohy dokumentu), aby bylo možno testovat použití této funkce přímo.

Na druhou stranu existují i důvody, proč funkci využít. Prodleva vložená do funkce není příliš velká na to, aby se nedala použít na hesla s kratší délkou. Obzvláště, zkrátí-li útočník zdrojovou abecedu. Útok hrubou silou je možno provádět po celou dobu, kdy je cílový stroj zapnutý a využívat tak pouze prostředky tohoto stroje. Je možno ukládat aktuální pozici, kterou lze za pomoci výpočtu obnovit i po restartu. Jako výhodné se také jeví užití slovníkové metody spolu s touto funkcí.

3.6.1.3 Komunikace

Vychází-li autor z faktu, že se úspěšně podařilo malware spustit na cílovém stroji, je požadováno, aby se malware napojil na server útočníka a teprve odtud přijímal příkazy k další práci.

Komunikace se váže na celý malware, proto je probírána nyní. Při implementaci se musí předem vědět, která ze stran vystupuje v roli serveru a která klienta. Zde je malware klient, který se pojí na server útočníka. Směr komunikace je zvolen s ohledem na fakta popisována v teoretické části.

V reálném světě komunikuje malware přes internet, proto jsou i v rámci testování využity technologie a protokoly sítě internet. Tedy TCP/IP. Komunikace probíhá přes sockety, jejichž podpora je implementována přímo v systému Windows. Sic jsou užity IP adresy virtuálních strojů, i přes to je ale projekt koncipován tak, že přes internet dokáže komunikovat. Stačí k tomu pouze IP adresy změnit. Je kladen požadavek na spolehlivost spojení - tedy aby každá zpráva

byla v pořádku doručena, proto je užit protokol TCP.

V systémech Windows je zajištěn přístup k socketům přes rozhraní Winsock. Aktuálně se užívá rozhraní verze 2[34], proto je projekt odvíjen od této verze.

Rozhraní nabízí potřebné funkce a struktury k vytvoření socketu, navázání spojení a k následnému čtení a zápisu. Ze všeho nejdříve je potřeba inicializovat strukturu "WSADATA", která přechovává informace o implementaci Windows Sockets. Před každým spojením musí být také definována cílová IP adresa, či doménové jméno, port a užitý protokol. Tyto údaje se zapisují do struktury "sockaddr_in". Je ovšem potřeba si dát pozor na řazení bytů dle rozdělení "little-endian" a "big-endian". Proto se užívá funkce "htons", která převede byty do pořadí užívaného na síti.

Samotný socket se pak definuje funkcí "socket" a ve funkci "setsockopt" se nastaví potřebné parametry. Nyní je již socket připraven k navázání spojení. To se provede funkcí "connect", která vrací chybovou hodnotu v případě selhání. Při neúspěchu lze konstantu chyby vyvolat funkcí "WSAGetLastError" a na stránkách dokumentace se programátor dozví bližší informace o způsobené chybě.

Je-li socket připojen, může z něj program číst a zapisovat do něj. Funkce "recv" umožní ze socketu číst, funkce "send" do socketu zapisovat. Obě funkce vrací počet bytů. V programu je potřeba ošetřit několik situací. Funkce "recv" stále přijímá data, ale neinterpretuje jejich vnitřní reprezentaci. To znamená, že odešle-li uživatel řetězec ukončený novým řádkem, funkce jej nerozpozná a bude dále číst. Je tedy potřeba řešit přijímání a ukončování řetězců (jednotlivých zpráv). Při odesílání se naopak musí kontrolovat, zda se odeslal celý řetězec a odesílat data, dokud se nerovná počet bytů odeslaných s délkou řetězce. Při odeslání každé ucelené zprávy se na její konec musí vložit ukončovací řetězec, který přijímající socket na vzdálené straně filtruje.

Programátor také nesmí zapomenout, že operace čtení i zápisu jsou operacemi blokujícími. Lze využít i neblokující verze těchto funkcí, ale v rámci lepší kontroly nad prováděním kódu je v projektu využito blokujících verzí, kdy každá z funkcí běží v samostatném vlákně.

Malware se tedy připojí k serveru a vyčkává na příkazy, které jsou vzdáleně volány.

3.6.1.4 Protokol

V rámci komunikace musí být užit i komunikační protokol, kterému budou obě strany rozumět. Existují mnohé, ale v projektu byl z důvodu specifických požadavků vytvořen protokol vlastní. Skládá se z parametrů oddělených mezerou. Povinné jsou dva úvodní, kdy první definuje identifikátor požadované úlohy, druhý definuje typ prováděné úlohy. Další parametry se liší dle požadované úlohy. Identifikátorem úlohy je číslo vyšší než nula. Nula slouží pro komunikaci se serverem. Každý typ úlohy je označován písmenem malé abecedy - konkrétní vysvětlení bude podáno v kapitole "akce". Je-li po klientovi požadována odpověď, zašle se jako typ úlohy 'a'. Chybové zprávy jsou zasílány protokolem jako typ úlohy 'e'. Každá zpráva protokolu je ukončována řetězcem "\r\n".

3.6.1.5 Akce

Malware provede po aktivaci pouze ty akce, které jsou popisovány v kapitole "po spuštění". O další odvíjení scénáře se stará útočník. Jedná-li se o časově náročnější úlohu, je implementováno její spuštění v novém vlákně.

- Správa úloh - Malware obsahuje vlastní správu úloh, která útočníkovi dává přehled o běžících úkonech. Úlohy mohou být ukončovány a nebo zjišťován jejich aktuální stav. Každá z úloh obsahuje informace o jednom z těchto stavů: neaktivní, aktivní, pouze pro čtení, ukončená, chybně ukončená. Také ukládá informace specifické pro zvolenou úlohu - výpis, počet spojení, chybovou hlášku. Pro využití správce úloh je protokol definován následně: parametr prováděné úlohy je roven 't', následovaný identifikátorem požadované úlohy. Posledním parametrem je 'i' v případě, že útočník požaduje informace o úloze, či 's', chce-li úlohu zastavit.
- Ukončení - Malware je na vzdáleném stroji odpojen a ukončen. Stačí pouze využít parametr 'd'.
- Zahájení DoS útoku - Program je připraven k zahájení DoS útoku. Každá žádost o DoS útok vytvoří vlákno, které následně spravuje dalších 100 vláken, které se připojují na definovanou adresu a port. K tomuto typu útoku se hodí využít konkrétně SYNflood útok. Komunikace je zprostředkována přes raw sockety, ovšem z důvodů limitace systému to nebylo v projektu možné využít. Spojení probíhá přes protokol TCP. Komunikační protokol vypadá následovně: jako typ úlohy se definuje 'r' a následují parametry "dos", vzdálená ip adresa, vzdálený port. Zastavení útoku se provede přes správce úloh.
- Spuštění procesů - Útočník je standardně limitován pouze na to, co je v aplikaci implementováno. K rozšíření funkcí proto malware podporuje spouštění systémových procesů. Pomocí volání WinAPI se aktivuje vzdálený proces a na malware se napojí jeho výstupní proud, který umožňuje komunikaci s útočníkem. Tímto způsobem může útočník spustit libovolný systémový proces, ke kterému má přihlášený uživatel oprávnění. Ukončení procesu lze provést tímto příkazem: "taskkill /im proces", kde proces zastupuje jméno procesu.

3.6.1.6 Penetrace a šíření

Tato kapitola bude rozebírána pouze teoreticky, jelikož šíření malware je ilegální činností. Existují množství způsobů, kterými je možné malware šířit - rozličné způsoby penetrace a šíření jsou popsány v teoretické části. Koncept projektu je směřován na zasažení co možná největšího počtu uživatelů. Je dobré si uvědomit, že nemusí být zvolena jen jedna cesta. Útočník se může malware pokusit rozšířit vícero způsoby, ovšem k efektivitě bude zapotřebí užít zdroje a komunikační kanály internetu.

Lze využít elektronické pošty a rozesílání SPAMu. Útočník vytvoří robota, který bude procházet webové stránky a z nich filtrovat adresy. Může také využít různých databází, které na internetu nalezne, v neposlední řadě i naprogramovat malware způsobem, kdy odešle sám sebe přes emailové klienta na všechny dostupné kontakty. Úspěšnost tohoto šíření je diskutabilní, ovšem z historie je patrné, že se různé viry masově rozšířili právě tímto způsobem (viz kapitola "Počítačová kriminalita"). Útočník musí brát v potaz, že by měl uživatele přesvědčit, aby infikovaný soubor otevřel. Proto jej vhodné jej maskovat za jiný spustitelný soubor a lákavě upravit text zprávy.

Další možností je využití parazitického připojení na jiný soubor. Aby se malware aktivoval, je požadována změna PE hlavičky u původního spustitelného souboru. Po spuštění bude aktivován malware, který vykoná svou práci - uloží se v systému, aktivuje svou kopii a předá řízení původnímu programu. V tomto případě spadá malware do kategorie trojských koní. Nastává otázka, jak dostat takovýto soubor do oběhu. Předpokládá se, že pachatel nemá přístup k oficiálním distribučním kanálům legálního software. Útočník se tedy zaměří na alternativní kanály, jako jsou diskuzní fóra, P2P sítě, torrentové sítě a podobné. Zde zveřejní svůj napadený soubor a vyčkává na jeho stažení ostatními uživateli. Hostitelským programem často bývá právě komerční software nebo produkt umožňující ilegální aktivaci komerčního software. Z výzkumu Business Software Alliance[33] vyplývá, že až 57% populace využívá nelegální software. Tedy právě ten software, který je stažen z alternativních zdrojů a ilegálně aktivován. Podaří-li se útočníkovi napadnout žádaný produkt, může velmi zvýšit své šance na sestavení botnet sítě.

3.6.2 Implementace serverové části

Server je nedílnou součástí projektu, jelikož zastřešuje fungování celého infrastruktury. Umožňuje napojení jak klientů(malware), tak administrátorských konzolí. V pomyslné rovině budou tyto administrátorské přístupy pronajímány klientům, kteří budou mít zájem síť využít ke svým účelům.

3.6.2.1 Principy fungování

Chce-li být server přístupný z internetu, je potřeba na něj nasměrovat dva porty a povolit komunikaci skrze TCP. Samozřejmostí je tedy veřejná IP adresa. Po spuštění začne server naslouchat na obou portech. Na jeden port se napojují stroje infikované malware, na port další správci sítě a zákazníci.

Každé příchozí spojení je identifikováno dle připojeného portu a systémem je přiděleno oprávnění. Je vytvořena struktura, která uchovává informace o každém z připojení, včetně socketů, vláken a aktuálního stavu. Při každém spojení se vytvoří dvě nová vlákna. Jedno stále přijímá zprávy od klienta, druhému je předáno zpracování zprávy, vykonání akcí a následný výpis. Nově připojený klient také obdrží svůj unikátní identifikátor, kterým je nezbytný k provázání komunikace mezi malware a administrátorskou konzolí. Pro komunikace s touto konzolí byl rovněž vytvořen protokol (popisován později). Každé nové příchozí spojení od

administrátora či zákazníka musí být ověřeno pomocí jména a hesla. Po třech chybných pokusech je takový klient odpojen. Mezi každým pokus o ověření je záměrně vložena prodleva a to z důvodu obrany proti útoku hrubou silou.

Jsou-li připojeny stroje pomocí malware, může je administrátor vzdáleně ovládat přes konzoli. Každý požadavek od administrátora je zaslán nejdříve na server, který ověří dostupnost klienta a také vygeneruje pro požadavek jedinečný identifikátor. Zpráva spolu s identifikátorem je zaslána na požadovaný stroj, který zprávu zpracuje, vykoná požadované akce a zašle odpověď na server spolu s původním identifikátorem. Server schraňuje identifikátory zpráv a zároveň je váže na administrátorské identifikátory. Podle této vazby se určí, na které připojení se má zaslat odpověď od malware. Server odděluje jednotlivé zprávy od různých správcovských konzolí. Při výpisu akcí se tedy vypíší pouze ty, které spravuje aktuální administrátor.

Server také zaznamenává důležité části komunikace, včetně času vytvoření.

3.6.2.2 Komunikace

Jak již bylo řečeno, server bude naslouchat na dvou portech. U obou připojení je požadováno, aby se jednalo o spojení spolehlivého charakteru. Bude tedy využit protokol spojení TCP. Server je implementován pro systémy Linux, bude tedy užívána Linuxová verze socketů. Nejdříve je zapotřebí nastavit parametry spojení. K tomu je vyhrazena struktura "sockaddr_in", která definuje IP adresu, na které server naslouchá a také port. Socket se pak vytvoří funkcí "socket". Od této části se již liší užití socketů u klienta a na serveru. Klient spojení vytváří, server na něj naslouchá. Následuje svázání socketu s předdefinovanými parametry - to zajišťuje funkce "bind". Dále pak samotné naslouchání přes funkci "listen" a konečně přijímání nových spojení díky funkci "accept". Přijímání požadavku o spojení funguje takovým způsobem, že server čeká na žádost a to konkrétně u funkce "accept" (ta je tedy blokující), která vrátí nový socket, na který je připojen klient. V programu se pak klientský socket ukládá do struktury, která je pro něj vyhrazena. Aby bylo odbavení požadavku co nejrychlejší, jsou vláknem, které přijímá spojení, provedeny jen nejnutnější operace a obsluha připojeného klienta je předána do jiného vlákna. Vlákno přijímající nová spojení se vrátí k funkci "accept" a čeká na dalšího klienta.

Jsou vytvořeny dvě vlákna pro klienta. První slouží ke čtení z klientského socketu a další k vykonávání akcí a zápisu.

3.6.2.3 Protokol

Server obsahuje dva komunikační protokoly. Oba jsou vytvořeny z vyplývajících požadavků tohoto projektu. Jeden slouží ke komunikaci s malware (byl již popisován výše), druhý ke komunikaci s administrátorskou konzolí. Druhý zmiňovaný bude popisován touto kapitolou.

Jako administrátorská konzole poslouží program telnet. Po napojení je zapotřebí autentifikace. Server vyzve k zadání přihlašovacího jména a hesla. Po úspěšném ověření se administrátor dostává do systému a může zasílat příkazy. Příkaz se skládá z nejméně dvou požadovaných parametrů. První identifikuje klienta, druhý operaci. Klienti jsou označováni číslem větším než nula. Číslo nula slouží ke komunikaci se serverem. Na server je možno zaslat parametr 'l', který vypíše připojené klienty(malware) nebo parametr 't', vypisující úlohy aktuálního uživatele. Parametr 'd' odpojí relaci. V případě chyby zasílá server zprávu označenou jako 'e', odpověď jako 'a'. Server vkládá jako první parametr odpovědi identifikátor 0. Jedná-li se o odpověď od klienta, je vložen identifikátor klienta. Ostatní příkazy se řídí pravidly užívanými ke komunikaci s malware. Protokol odděluje jednotlivé zprávy řetězcem "\r\n".

4 Dosažené výsledky a studie

4.1 Testování

Testování je velmi potřebné při vývoji každého systému a v každém případě je nutné jej provádět již během implementace. Testování celé architektury bylo prováděno na všech vymezených operačních systémech a to jak během implementace, tak po dokončení programátorských prací. Z otestovaných výsledků byly usuzovány další kroky při implementaci, či opravy nesprávně fungujících částí. Každá zde popisovaná funkcionalita byla otestována, zda skutečně plní svůj účel. V případě že výsledek nebyl správný, byl nalezen takový způsob, který požadovaného výsledku dosáhl. Bylo potřeba také otestovat, jak se malware chová v různých prostředích. Některé funkce musely být poupraveny způsobem, aby vykazovaly stejné známky chování jak ve starším systému Windows XP, tak v novějších Windows 7 a Windows 8. Toho bylo docíleno větvením programu dle zjištěné verze systému.

4.2 Životní cyklus a možné následky rozšíření

Bylo popisováno, jakým způsobem byl malware vytvářen a testován. Nyní budou probírány různé scénáře, které mohou nastat v případě, že se malware útočníkovi podaří dostatečně rozšířit.

Každý počítač nakažený tímto malware je možno vzdáleně ovládat a to dle libosti útočníka. Záleží na jeho plánech, které s nakaženým systémem zamýšlí. Stroj oběti tak může být využit k ilegálním činnostem, dalšímu šíření malware či provádění útoku. Právě k provádění DoS útoku směřoval vývoj malware. Pachatel ovšem může získat i přístup k citlivým datům, či přihlašovacím údajům. Odchytávání hesel není v projektu implementováno, ale je možné malware o tuto funkci rozšířit. Proto je potřeba co možná nejvíce předcházet těmto scénářům pomocí dobrého zabezpečení systému a zdravého úsudku. Je potřeba eliminovat všechna rizika, která mohou spustit nebezpečný kód.

Další ze scénářů je rozebírán z pohledu společnosti, na kterou byl použit DoS útok. Záleží na konkrétní společnosti, jaké služby přes internet nabízí a jak moc je na nich založen jejich profit. Jsou-li právě internetové služby jejich jádrem, mohou se škody způsobené výpadkem služeb velmi prodražit.

4.3 Budoucí vylepšení a vývoj

Dá se říci, že většinu programů je možné dále vylepšovat a přidávat nové funkce. To stejné platí i v tomto projektu. Malware i server splnili očekávání projektu, přinesli funkce navíc, ale i přes to je mnoho způsobů, jakými je možné je rozšířit.

Server je možno napojit na databázi uživatelů a využívat různé možnosti autorizace pro různé operace a různé uživatele. Dále je určitě vhodné implementovat dávkové příkazy, které umožní zaslat útočníkem jen jedinou zprávu, která bude distribuována na všechny systémy. Distribuční listy mohou být také ukládány v databázi. Pro větší administrátorský komfort je možno implementovat grafickou aplikaci.

Mnoho způsobů vylepšení může být aplikováno i na samotný malware. Může být ještě sofistikovanějším, než je nyní. Je možno rozšiřovat množství funkcí, které bude malware plnit, implementovat nové způsoby, které zajistí zvýšení oprávnění, lépe jej skrýt před uživatelem. Ale především také nalézat nové způsoby, kterými jej šířit. Pro útočníka by bylo nejvýhodnější, kdyby mohl malware využít vzdálené zranitelnosti. To je ovšem v praxi velmi málo proveditelné.

Budoucí vývoj je zajištěn především díky zájmu autora o problematiku počítačové bezpečnosti. Následné kroky budou směrovány aktuálními trendy v oblasti bezpečnosti, ale také tématy, které autora zaujmou.

4.4 Detekce malware

V rámci projektu byly testovány i způsoby detekce. Detekce je dle kapitoly "ukrytí" rozdělena na dva možné případy: detekce uživatelem a detekce bezpečnostními systémy. Kapitola se zabývá detekcí již běžícího malware. Nebude tedy uvažováno, že malware ještě nebyl spuštěn.

Z pohledu uživatele nejsou na první pohled patrné žádné známky chování škodlivého kódu. Uživatel se tedy musí zaměřit hlouběji na systém. Ve správci úloh je úloha zobrazena jako "svchost.exe", což je jméno systémového procesu a nemusí být na první pohled patrné, že se jedná právě o škodlivý kód. Znalý uživatel ovšem zjistí, že proces by neměl běžet pod jeho uživatelským účtem, ale pod systémovým. Další, již patrnější, známkou je záznam v registru uchovávající programy, které jsou spouštěny po přihlášení. Opět záleží na uživateli, zda program bude považovat za systémový. Asi nejpatrnějším znakem bude možné zobrazení okna brány firewall. Záleží na okolnostech, zda k zobrazení okna dojde. I přes to jej uživatel může povolit, jelikož si bude myslet, že se jedná o proces systémový.

Malware je nově implementován a proto je možné, že jej bezpečnostní programy nerozpoznají jako hrozbu. Toto tvrzení bylo autorem otestováno. Z celkového počtu 52 bezpečnostních programů jej rozpoznalo programů 8. Mezi úspěšné zástupce patří: AVG, F-Secure, McAfee, BitDefender a další. Nejčastěji byl malware identifikován jako "Gen:Trojan.Heur.FU.GGW@aGbW99j", dále pak "Win32/DH{D1VEICI}" a "Heuristic.LooksLike.Win32.Suspicious.J!89". Je patrné, že bezpečností software užily heuristickou analýzu. To je taková metoda, která je založena na analýze chování programů. Metoda se snaží nalézt takové kódy, které nejsou v databázi antivirového programu, ale svým chováním škodlivý kód připomínají.

4.5 Prevence a doporučení

Práce odhaluje mnoho postupů, které jsou často užívány útočníky k proniknutí do systémů, ukrytí hrozby, šíření, či jiné. Z pohledu zabezpečení je důležité znát tyto způsoby, aby se uživatel či správce systému mohl proti jejich užití bránit.

Z textů lze vyvodit několik závěrů: nejslabším článkem v oblasti zabezpečení se může stát právě lidský faktor. Především pak neznalost, nedbalost, či nadměrná důvěřivost.

Toto platí především pro uživatele. Z výzkumů je patrné, že uživatelé se sami stávají terči malware dobrovolným užíváním ilegálně získaného software.

Kapitola "sociální inženýrství" popisuje různé útoky na lidské slabiny. Je proto potřeba si informace co možná nejlépe ověřovat, nikdy nesdílet a nevydávat své přihlašovací údaje. Důležité systémy chránit jinými přihlašovacími údaji a neregistrovat se na nedůvěryhodných stránkách, které mohou sbírat uživatelské údaje.

V dokumentu je také popisován útok hrubou silou. Je vhodné volit hesla, která jsou dostatečně silná na to, aby provedení útoku nebylo za normálních podmínek úspěšné.

Samozřejmě jsou hrozbou i aplikace, které obsahují bezpečnostní slabiny. To je ovšem riziko, s kterým musí uživatel při užívání takovýchto aplikací počítat. Uživatel může přispět k vyššímu zabezpečení pravidelnými aktualizacemi.

Z práce je také čitelné, že implementovaný malware byl zachycen některými bezpečnostními programy. Je vhodné jejich užití, zvláště i brány firewall, jelikož právě ta se v projektu jevila jako největší překážka pro útočníka.

Určitá míra paranoie je tedy na místě - při podivném chování se vyplatí systém analyzovat a zjistit tak důvody těchto problémů. Z hlediska obrany je důležitým prvkem záznam (log) události. Právě tyto záznamy mohou dopomoci k identifikaci problému.

4.6 Hodnocení

Teoretická část se postupně zabývá různými aspekty, postupy, technickým řešením, či dovednostmi, které jsou potřebné pro vytvoření malware. Autor nastudoval všechna potřebná témata a z těchto nově, či předem získaných zkušeností vytvořil práci odpovídajícího charakteru. Části na sebe vzájemně navazují logickým uspořádáním, kdy jsou nejdříve probírány potřebné prerekvizity a technologie, dále je popisováno postupné vytvoření malware až po jeho dokončení a následné šíření.

V praktické části jsou vymezeny konkrétní požadavky na malware. Byla definována architektura, která se skládá z klientské a serverové části. Obě části byly implementovány dle požadavků. Postupně je probírána implementace jednotlivých částí, popisovány způsoby, jakými jsou užity jednotlivé technologie a vyřešeny překážky k dosažení cíle. Praktická část slouží i jako dokumentace k systému, jelikož popisuje i způsob, jakým malware i server fungují

a jak je možné přistupovat k jejich ovládání.

Malware splňuje charakter škodlivého kódu, jelikož před uživatelem maskuje svůj chod různými způsoby a plní příkazy definované útočníkem. Malware zkopíruje sám sebe do systémových složek uživatele, vytvoří hodnotu registru a zajistí tak spouštění při každém přihlášení uživatele. Při běhu není zobrazeno žádné okno, malware vytvoří pravidlo brány firewall, aby mohl komunikovat se serverem a odtud přijímat požadavky. Malware implementuje metody k provedení DoS útoku, tím tedy splňuje i požadavek na fiktivní scénář vytvoření sítě botnet. Implementace tak odráží všechny prvky botnet architektury užívané pachateli.

Autor díky studiu, implementaci a rozmýšlení nad tématem získal ucelený pohled na problematiku bezpečnosti hardware, sítí, systému, ale taky o možnostech škodlivého kódu. Jelikož se vžil do role útočníka, mohl uvažovat způsoby, které se snaží najít i ty nejméně patrná zneužitelná místa. Všechny tyto poznatky se dají lehce převést na roli správce systému a vytvořit díky nim potřebná bezpečnostní opatření. Zjištěná fakta budou autorem užita ke zlepšení bezpečnosti spravovaných systémů a implementovaných aplikací.

Byly probrány všechny potřebné předpoklady, principy, technická řešení, teoretická i praktická východiska pro tvorbu malware. Projekt je v pořádku implementován - jak malware, tak server jsou funkční a komunikace mezi nimi probíhá dle očekávání. V práci byly splněny předem definované požadavky a to i nad jejich rámec.

5 Závěr

Práce podala ucelené informace o možnostech počítačových hrozeb, jejich šíření, ale následně i obraně proti těmto hrozbám. Je patrné, že existují méně, i více sofistikované hrozby.

Autorem byla implementována architektura, která poukazuje na dovednosti, které musí útočník k tvorbě takového malware ovládat. Je patrné, že techniky zde užité patří mezi pokročilé možnosti programování a proto se nemusí uživatel obávat, že malware bude schopen každý vytvořit. Jsou ovšem různé možnosti, které i začínajícím programátorům umožní vytvořit takové kódy, které vykonají nechtěnou akci. Ať již v důsledku svého jednání, či pouhou neopatrností, můžete narazit na škodlivé kódy, které využijí Váš přístroj a data nechtěným způsobem. S velkou pravděpodobností to ovšem může být právě Vaše chyba, která umožní jejich úspěšné vykonání. Je důležité si uvědomovat kroky svého jednání v digitálním světě a brát na zřetel, že ne každý přichází s čistými úmysly. Byly popsány různé metody, které co nejvíce eliminují možnosti vykonání škodlivého kódu. Dodržováním těchto zásad si ulehčíte hodně starostí a někdy i financí.

Čas od času se v televizi objeví film, který představuje hackery jako ty, kteří se bez potíží napojují na komunikační síť, vypnou elektrické rozvody celého města, ovládají ukradené satelity, mění svou totožnost, či vykrádají ve velkém bankovní konta.

Tyto scénáře jsou často jen fikcí snažící se vzbudit v pozorovateli pocity úžasu. Reálné napadení takového systému je buď skrze internetové spojení nemožné, nebo opravdu velmi náročné. Jak dokazuje historie, nedošlo k takovým útokům, které by znehodnotily tok peněz, či jiné systémy v globálním měřítku. Každý den jsou ovšem prováděny útoky, které zahlcují internetové servery, snaží se získat uživatelská data, osobní údaje a také čísla platebních karet. Je dobré si uvědomit, že v případech, kdy útočníkovi sami nepomůžeme svým jednáním, bude pro něj velmi těžké, ne-li nemožné, tyto informace získat. Metody útočníků jsou stále sofistikovanější, naštěstí i bezpečnostní programy dělají systémy stále nedobytnějšími.

Počítačová bezpečnost je stále aktuální téma. Bude aktuální do té doby, než budou všechny systémy buď zničeny, nebo ochráněny tak, že útok na ně již nebude možný.

Literatura

1. ERICKSON, Jon. *Hacking: umění exploitace*. 2., upr. a dopl. vyd. Překlad Jan Pokorný. Brno: Zoner Press, 2009, 544 s. ISBN 978-80-7413-022-9.

2. SELECKÝ, Matúš. *Penetrační testy a exploitace*. 1. vyd. Brno: Computer Press, 2012, 303 s. ISBN 978-80-251-3752-9.

3. HARRIS, Shon, Allen HARPER, Chris EAGLE, Jonathan NESS a Michael LESTER. *Hacking: manuál hackera*. 1. vyd. Praha: Grada, 2008, 399 s. ISBN 978-80-247-1346-5.

4. SZOR, Peter. *Počítačové viry: analýza útoku a obrana*. Vyd. 1. Brno: Zoner Press, 2006, 608 s. ISBN 80-868-1504-8.

5. GRAVES, Kimberly. *CEH: certified ethical hacker study guide*. Indianapolis, Ind.: Wiley Pub., c2010, xxxvii, 392 p. ISBN 978-0-470-52520-3.

6. HEINIGE, Karel. *Viry a počítače*. Praha: Knihy iDNES, 2001. ISBN 80-86097-74-9.

7. *Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code*. Indianapolis: Wiley, c2011, xxvi, 716 s. ISBN 978-0-470-61303-0.

8. JAMES, Lance. *Phishing bez záhad*. 1. vyd. Praha: Grada, 2007, 281 s. ISBN 978-80-247-1766-1.

9. CLARKE, Nathan. *Proceedings of the Sixth International Symposium on Human Aspects of Information Security: Crete, Greece, 6-8 June 2012*. United Kingdom: University do Plymouth, 2007, iv, 216 pages. ISBN 978-184-1023-175.

10. MAREK, Rudolf. *Učíme se programovat v jazyce Assembler pro PC*. Vyd. 1. Brno: Computer Press, 2003, 228 s. ISBN 80-722-6843-0.

11. MERHAUT, Filip a Ivan ZELINKA. *Počítačové viry a bezpečnost*. Ostrava, 2012. Dostupné z: http://arg.vsb.cz/data/Vyuka/PVBPS.pdf

12. HÁK, Igor. *Moderní a počítačové viry*. třetí. Dostupné z: http://viry.cz/download/kniha.pdf

13. ZULFIKAR, Stamm SID a MARKUS. *Drive-by pharming*. 2006. Dostupné z: http://research.sidstamm.com/papers/driveby-pharming.pdf

14. Česká republika. Sbírka zákonů. In: *40. Zákon trestní zákoník*. 2009, 11. Dostupné z: http://business.center.cz/business/pravo/zakony/trestni-zakonik/

15. LÁTAL, Ivo. Počítačová (informační) kriminalita a úloha policisty při jejím řesení. *Policista* [online]. 1998, č. 3 [cit. 2014-05-06]. Dostupné z: http://www.scritub.com/limba/ceha-slovaca/Potaov-informan-kriminalita-a-1513463.php

16. Computer viruses hit one million. *Http://www.bbc.co.uk/* [online]. 2014, aktualizováno 6.5.2014 [cit. 2014-05-06]. Dostupné z: http://news.bbc.co.uk/2/hi/technology/7340315.stm

17. Top 5 Computer Viruses Of All Time. *Http://uk.norton.com/* [online]. ©1995 - 2014 [cit. 2014-05-06]. Dostupné z: http://uk.norton.com/top-5-viruses/promo

18. Why People Create Computer Viruses?. In: *Http://www.nortonantiviruscenter.com/* [online]. © 2007-2013 [cit. 2014-05-06]. Dostupné z: http://www.nortonantiviruscenter.com/security-resource-center/why-people-create-computer-viruses.html

19. WHITTY. Why do People Create Computer Viruses?. In: *Http://www.technibble.com/* [online]. © 2006-2014 [cit. 2014-05-06]. Dostupné z: http://www.technibble.com/why-do-people-create-computer-viruses/

20. Slovník pojmů. In: *Http://www-old.gts.cz/gtsbezpecnyinternet/cs/* [online]. © 2013 [cit. 2014-05-06]. Dostupné z: http://www-old.gts.cz/gtsbezpecnyinternet/cs/napoveda/slovnik-pojmu/

21. *Speedguide.net* [online]. © 1998-2014 [cit. 2014-05-06]. Dostupné z: http://www.speedguide.net/ports.php

22. List of TCP and UDP port numbers. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001-2014, 6.5.2014 [cit. 2014-05-06]. Dostupné z: http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers

23. Backdoor found in D-Link router firmware code. In: *InfoWorld* [online]. Palo Alto, CA: InfoWorld, 1980- [cit. 2014-05-06]. Dostupné z: http://www.infoworld.com/d/security/backdoor-found-in-d-link-router-firmware-code-228725

24. KAMKAR. *NAT Pinning: Penetrating routers and firewalls from a web page (forcing router to port forward)* [online]. 2010 [cit. 6.5.2014]. Dostupné z:

http://samy.pl/natpin/

25. *JonDonym: IP check* [online]. [cit. 6.5.2014]. Dostupné z: http://ip-check.info/?lang=en

26. anopticlick. *Panopticlick* [online]. 2010 [cit. 2014-05-07]. Dostupné z: https://panopticlick.eff.org/

27. Check Point Survey Reveals Nearly Half of Enterprises Are Victims of Social Engineering: Social engineering attacks can cost businesses more than $100,000 per incident, emphasizing the importance of better security and user awareness. *Http://www.checkpoint.com/* [online]. ©2014 [cit. 2014-05-07]. Dostupné z: http://www.checkpoint.com/press/2011/092111-enterprises-victims-social-engineering.html

28. VISUAL STUDIO .NET 2003. *Programming Languages* [online]. [cit. 6.5.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa292164%28v=vs.71%29.aspx

29. *Windows API* [online]. [cit. 6.5.2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/ff818516%28v=vs.85%29.aspx

30. PARKER, Don. Windows NTFS Alternate Data Streams. In: *Symantec* [online]. ©1995 - 2014 [cit. 2014-05-07]. Dostupné z: http://www.symantec.com/connect/articles/windows-ntfs-alternate-data-streams

31. The Exploit Database. *Exploit Database* [online]. 2014, 2.5.2014 [cit. 2014-05-07]. Dostupné z: http://www.exploit-db.com/

32. Inj3ct0r. *1337day* [online]. © 2008-2014 [cit. 2014-05-07]. Dostupné z: http://1337day.com/

33. Realtime Web Analytics With no Sampling: Desktop Operating System Market Share. *NETMARKETSHARE* [online]. © 2006-2014 [cit. 2014-05-07]. Dostupné z: http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0&qpsp=182&qpnp=1&qptimeframe=M

34. Windows Sockets 2. In: *Windows* [online]. © 2014 [cit. 2014-05-07]. Dostupné z: http://msdn.microsoft.com/en-us/library/windows/desktop/ms740673%28v=vs.85%29.aspx

Seznam příloh

Příloha 1: Příloha na CD - zdrojové kódy.